MSP430F149 学习之端口输入输出实验

1、代码：

```
#include "io430.h"
int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  //端口初始化
  P1SEL=0X00;//P1 端口做为 I/O 端口
  P1DIR=0X00;//P1 端口做为输入端口
  P5SEL=0X00;//P5 端口做为 I/O 端口
  P5DIR=0XFF;//P5 端口做为输出端口

  //核心部分
  for(;;)
  {
    if((P1IN&BIT0)==0)
      P5OUT&=~BIT0;
    else
      P5OUT|=BIT0;
    if((P1IN&BIT1)==0)
      P5OUT&=~BIT1;
    else
      P5OUT|=BIT1;
    if((P1IN&BIT2)==0)
      P5OUT&=~BIT2;
    else
      P5OUT|=BIT2;
    if((P1IN&BIT3)==0)
      P5OUT&=~BIT3;
    else
      P5OUT|=BIT3;
  }
  //return 0;
}
```

2、总结：

输出引脚定义：

PmOUT&=～BITn　　Pm.n 输出低电平

PmOUT|=BITn　　Pm.n 输出高电平

输入引脚定义：

PmIN&BITn　　Pm.n

MSP430F149 学习之端口中断与端口输出实验

1、代码

```
#include "io430.h"
void delay(void)
{
   int i,j;
   for(i=0;i<100;i++)
      for(j=0;j<300;j++);
}

#pragma vector = PORT1_VECTOR
__interrupt void p1init(void)     //注意：interrupt 前的下划线是两条，不是一条!!
{
      if((P1IN&BIT0)==0)
         P5OUT&=~BIT0;

      if((P1IN&BIT1)==0)
         P5OUT&=~BIT1;
      if((P1IN&BIT2)==0)
         P5OUT&=~BIT2;
      if((P1IN&BIT3)==0)
         P5OUT&=~BIT3;

      delay();//使得发光 LED 稳定显示

      P1IFG=0;//中断标志位清零!!!
}
int main( void )
{
   // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;
   //端口初始化
   P1DIR|=0X00;   //P1.0、P1.1、P1.2、P1.3 为输入模式
   P1IE|=0X0F;     //P1.0、P1.1、P1.2、P1.3 允许中断
   P1IES|=0x0F;   //P1.0、P1.1、P1.2、P1.3 为下降沿触发
   P5DIR|=0X0F;   //P5.0、P5.1、P5.2、P5.3 为输出模式

   asm("eint");//打开总中断开关
   for(;;)
   {
      P5OUT=0XFF;
```

```
  }
  //return 0;
}
```

2、总结：

（1）、__interrupt void p1init(void)

interrupt 前的下划线是两条，不是一条!!

（2）、开启总中断方式： asm("eint");

（3）、注意中断标志位的清零。

（4）、中断的语法规则：

<1>#pragma vector=PORT1_VECTOR

PORT1_VECTOR 是中断向量，即此中断是为 PORT1_VECTOR 中断向量服务的。

<2>__interrupt void p1init(void){ } 中断函数声明

MSP430F149 学习之数码管显示

1、代码：

```
#include "io430.h"
#define Duan P6OUT                    //段选
#define DuanC_H P5OUT|=BIT7     //段选控制位 高电平
#define DuanC_L P5OUT&=~BIT7   //段选控制位 低电平
#define WeiA_H   P5OUT|=BIT6     //位选 A 高电平
#define WeiA_L   P5OUT&=~BIT6   //位选 A 低电平
#define WeiB_H   P5OUT|=BIT5     //位选 B 高电平
#define WeiB_L   P5OUT&=~BIT5   //位选 B 低电平
#define WeiC_H   P5OUT|=BIT4     //位选 C 高电平
#define WeiC_L   P5OUT&=~BIT4   //位选 C 低电平
//数码管编码表 0-9,a-f
const unsigned char
SMG[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71}
;
//const unsigned char
SMG[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e}
;
void SMG_Show(unsigned int duan,unsigned int wei);//数码管显示函数
void delay(void);//延时函数
int main( void )
{
  unsigned int i,j,m;

  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  //端口初始化
  P6DIR=0XFF; //P6 为输出模式
  P5DIR|=0XF0;//P5.4、P5.5、P5.6、P5.7 做为输出端口
```

```
    for(;;)
    {
      for(m=0;m<10000;m++)
      {
        j=1;
        for(i=0;i<8;i++,j++)
        {
          SMG_Show(i,j);
          delay();
        }
      }
      for(m=0;m<10000;m++)
      {
        j=1;
        for(i=8;i<16;i++,j++)
        {
          SMG_Show(i,j);
          delay();
        }
      }
    }
    //return 0;
}
void SMG_Show(unsigned int duan,unsigned int wei)
{

    DuanC_H;//段选控制打开
    Duan=SMG[duan];
    DuanC_L;//段选控制关闭


    switch(wei)
    {
      case 1:
        WeiA_H;WeiB_H;WeiC_H;break;

      case 2:
        WeiA_L;WeiB_H;WeiC_H;break;

      case 3:
        WeiA_H;WeiB_L;WeiC_H;break;

      case 4:
```

```
        WeiA_L;WeiB_L;WeiC_H;break;

    case 5:
        WeiA_H;WeiB_H;WeiC_L;break;

    case 6:
        WeiA_L;WeiB_H;WeiC_L;break;

    case 7:
        WeiA_H;WeiB_L;WeiC_L;break;

    case 8:
        WeiA_L;WeiB_L;WeiC_L;break;
  }
}
void delay(void)
{
  unsigned int i,j;
  for(i=0;i<19;i++)
    for(j=0;j<25;j++);
}
```

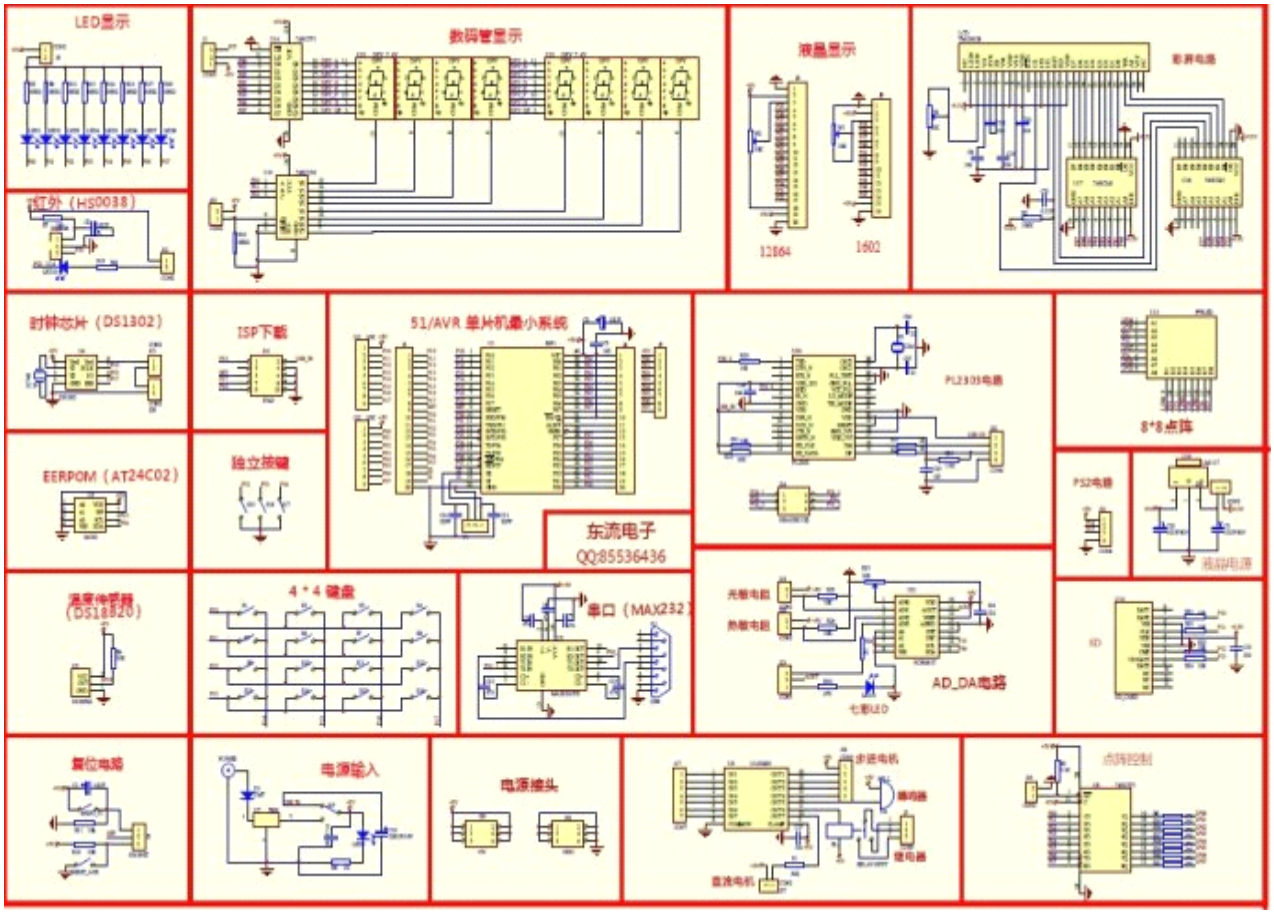
2、总结：

通过使用锁存器和译码器来节省单片机的 I/O 口资源，同时实现端口的复用。

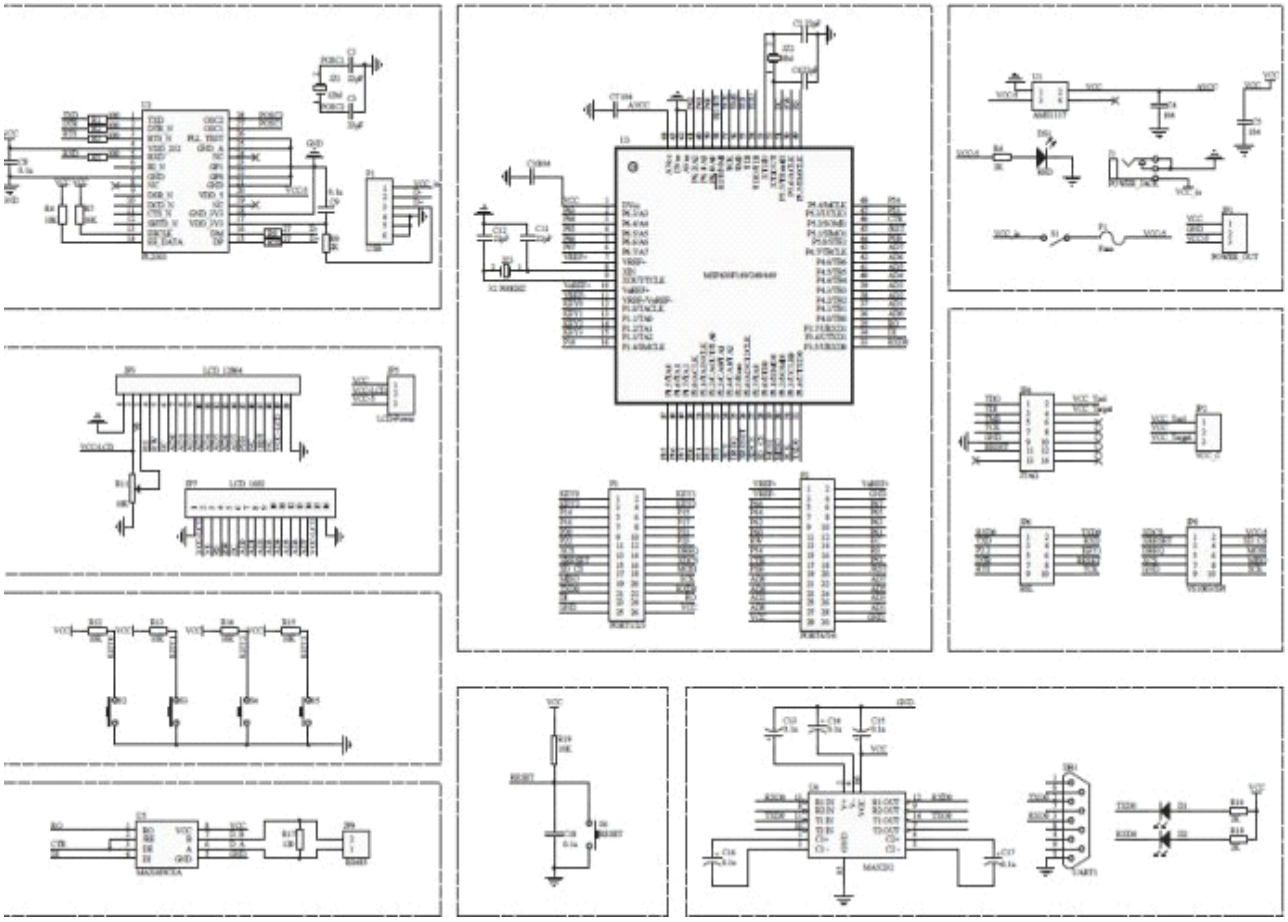
MSP430F149 学习之外设原理图和最小系统原理图

以前学习 51 时，买了一块开发板，现在学习 MSP430，为了节省银子，MSP430 就买了最小系统，外设全部使用 51 板上的资源。

MSP430 外设图：



MSP430 最小系统图：



MSP430F149 学习之 4*4 矩阵键盘与数码管显示

```
#include "io430.h"
/
int main( void )
{
   unsigned int KeyValue=16,KeyValueLast=0;

   // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;
   //端口初始化
   P6DIR=0XFF; //P6 为输出模式
   P5DIR|=0XF0;//P5.4、P5.5、P5.6、P5.7 做为输出端口

   for(;;)
   {
      KeyValue=KeyBoardScan();
      switch(KeyValue)
      {
         case 1: SMG_Show(1,1);KeyValueLast=KeyValue;KeyValue=0;break;
```

```
        case 2: SMG_Show(2,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 3: SMG_Show(3,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 4: SMG_Show(4,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 5: SMG_Show(5,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 6: SMG_Show(6,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 7: SMG_Show(7,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 8: SMG_Show(8,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 9: SMG_Show(9,1);KeyValueLast=KeyValue;KeyValue=0;break;
        case 10:
SMG_Show(1,2);delay();SMG_Show(0,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 11:
SMG_Show(1,2);delay();SMG_Show(1,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 12:
SMG_Show(1,2);delay();SMG_Show(2,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 13:
SMG_Show(1,2);delay();SMG_Show(3,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 14:
SMG_Show(1,2);delay();SMG_Show(4,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 15:
SMG_Show(1,2);delay();SMG_Show(5,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        case 16:
SMG_Show(1,2);delay();SMG_Show(6,1);delay();KeyValueLast=KeyValue;KeyValue=0;break;
        default:
          switch(KeyValueLast)
          {
                case 1: SMG_Show(1,1);break;
                case 2: SMG_Show(2,1);break;
                case 3: SMG_Show(3,1);break;
                case 4: SMG_Show(4,1);break;
                case 5: SMG_Show(5,1);break;
                case 6: SMG_Show(6,1);break;
                case 7: SMG_Show(7,1);break;
                case 8: SMG_Show(8,1);break;
                case 9: SMG_Show(9,1);break;
                case 10:   SMG_Show(1,2);delay();SMG_Show(0,1);delay();;break;
                case 11:   SMG_Show(1,2);delay();SMG_Show(1,1);delay();break;
                case 12:   SMG_Show(1,2);delay();SMG_Show(2,1);delay();break;
                case 13:   SMG_Show(1,2);delay();SMG_Show(3,1);delay();break;
                case 14:   SMG_Show(1,2);delay();SMG_Show(4,1);delay();break;
                case 15:   SMG_Show(1,2);delay();SMG_Show(5,1);delay();break;
                case 16:   SMG_Show(1,2);delay();SMG_Show(6,1);delay();break;
          }
    }
```

```
    }
    //return 0;
}

void SMG_Show(unsigned int duan,unsigned int wei)
{

    DuanC_H;//段选控制打开
    Duan=SMG[duan];
    DuanC_L;//段选控制关闭


    switch(wei)
    {
        case 1:
            WeiA_H;WeiB_H;WeiC_H;break;

        case 2:
            WeiA_L;WeiB_H;WeiC_H;break;

        case 3:
            WeiA_H;WeiB_L;WeiC_H;break;

        case 4:
            WeiA_L;WeiB_L;WeiC_H;break;

        case 5:
            WeiA_H;WeiB_H;WeiC_L;break;

        case 6:
            WeiA_L;WeiB_H;WeiC_L;break;

        case 7:
            WeiA_H;WeiB_L;WeiC_L;break;

        case 8:
            WeiA_L;WeiB_L;WeiC_L;break;
    }
}

void delay(void)
{
    unsigned int i;
    for(i=0;i<1800;i++);
```

```
}

unsigned int KeyBoardScan(void)
{
  //判断键盘状态
  P4DIR=0X0F;
  P4OUT=0X00;
  if((P4IN&0XF0)!=0XF0    &&    KeyBoardSign==1) return 0;
  else KeyBoardSign=0;

  //第一行扫描
  P4DIR=0X0F;
  P4OUT=0X0e;
  switch(P4IN&0XF0)
  {
     case 0xe0:    KeyBoardSign=1;return 1;
     case 0xd0:    KeyBoardSign=1;return 2;
     case 0xb0:    KeyBoardSign=1;return 3;
     case 0x70:    KeyBoardSign=1;return 4;
  }

  //第二行扫描
  P4DIR=0X0F;
  P4OUT=0X0d;
  switch(P4IN&0XF0)
  {
     case 0xe0:    KeyBoardSign=1;return 5;
     case 0xd0:    KeyBoardSign=1;return 6;
     case 0xb0:    KeyBoardSign=1;return 7;
     case 0x70:    KeyBoardSign=1;return 8;
  }

  //第三行扫描
  P4DIR=0X0F;
  P4OUT=0X0b;
  switch(P4IN&0XF0)
  {
     case 0xe0:    KeyBoardSign=1;return 9;
     case 0xd0:    KeyBoardSign=1;return 10;
     case 0xb0:    KeyBoardSign=1;return 11;
     case 0x70:    KeyBoardSign=1;return 12;
  }

  //第四行扫描
```

```
    P4DIR=0X0F;
    P4OUT=0X07;
    switch(P4IN&0XF0)
    {
       case 0xe0:    KeyBoardSign=1;return 13;
       case 0xd0:    KeyBoardSign=1;return 14;
       case 0xb0:    KeyBoardSign=1;return 15;
       case 0x70:    KeyBoardSign=1;return 16;
    }

    return 0;
}
```

MSP430F149 学习之 Nokia5110 显示

1、main.c:

```
#include    "io430.h"
#include    "Font_code(8x16).c"
#include    "Font_code(6x8).c"
#include    "picture.c"
#include    "Nokia5110.h"
unsigned char    *str1 = {"Welcome"};
unsigned char    *str2 = {"Nice to meet you!"};
unsigned char    *str3 = {"SKYWOLF"};
unsigned char
str4[]={0x80,0x70,0x00,0xFF,0x10,0x22,0xF2,0x92,0x92,0x92,0x92,0x92,0xFB,0x12,0x00,0x00,

0x00,0x00,0x00,0xFF,0x20,0x20,0x27,0x24,0x24,0x24,0x24,0x24,0x27,0x30,0x20,0x00};//恒
void Clock_Init(void);//时钟初始化
void delay_1us(void);//延时函数，延时时间 1us
void delay250ms(void);//延时 250ms
void delay1s(void);//延时 1s
void delay40ms(void);//延时 40ms
int main( void )
{
    unsigned char i=0,j=0,k=0;
    WDTCTL = WDTPW + WDTHOLD;//关闭看门狗
    Clock_Init();//8M 主频,125ns

    P2DIR=0XFF;

    LCD5110_Init();

    //静态显示英文字符
    LCD_prints_6x8(0,3,">>>Welcome<<<");
```

```
delay1s();
LCD_clr_scr();

//静态显示汉字
LCD_printch_16x16(0,1,str4);
delay1s();
LCD_clr_scr();

for(;;)
{
  //8X16 字体
  LCD_prints_8x16(0,0,str1);//静态显示
  LCD_prints_8x16(0,2,str3);//静态显示
  while(*str2)                    //动态显示
  {
    LCD_prints_8x16(0,1,str2++);
    i++;
    delay250ms();
  }
  str2-=i;
  i=0;

  LCD_clr_scr();

  //6X8 字体
  LCD_prints_6x8(0,0,str1);//静态显示
  LCD_prints_6x8(0,2,str3);//静态显示
  while(*str2)                    //动态显示
  {
    LCD_prints_6x8(0,1,str2++);
    i++;
    delay250ms();
  }
  str2-=i;
  i=0;
  LCD_clr_scr();

  //显示图片
  delay1s();
  LCD_picture_84x48(pic);
  delay1s();
  delay1s();
  LCD_clr_scr();
```

```
LCD_pos_picture(0, 0, 72, 14, pic1);
delay1s();
LCD_pos_picture(0, 2, 72, 28, pic2);
delay1s(); delay1s();
LCD_clr_scr();//清屏

//显示英文字符
for(i = 0; i < 14; i++) {LCD_printc_6x8(i, 2, str2[i]); delay250ms();}
LCD_printn_6x8(2, 3, 13140, 5);
delay1s();
LCD_prints_6x8(0, 4, ">--(*^_^*)--<");
delay1s();
LCD_clr_scr();

//显示屏相关测试
LCD_prints_6x8(0, 0, "Blank Test ");
delay1s();
LCD_show_blank; //空白测试
delay1s();
LCD_prints_6x8(0, 0, "Normal Test");
LCD_show_normal; //恢复正常
delay1s();
LCD_prints_6x8(0, 0, "Black Test ");
delay1s();
LCD_show_black; //全黑检测坏点
delay1s();
LCD_show_normal; //恢复正常
delay1s();
LCD_prints_6x8(0, 0, "Inverse Test");
LCD_show_inverse; //反色
delay1s();
LCD_prints_6x8(0, 0, "Normal again");
LCD_show_normal; //恢复正常
delay1s();
LCD_clr_scr(); //清屏

i = 0; j = 0; k =0;
LCD_prints_6x8(0, 5, "ASCII Code:");
while(k++ < 100) //ASCII 字符测试
{
   LCD_printc_6x8(i, j, k + 32);
   delay250ms();
   i++;
   if(i == 14) {i = 0; j++;}
```

```
            if(j == 5) j = 0;
            LCD_printn_6x8(11, 5, k + 32, 3);
        }
        delay1s();
        LCD_clr_scr();      //清屏
    }

    //return 0;
}
void Clock_Init(void)
{
    unsigned int i;
    P5DIR|=0X10;
    P5SEL|=0X10;
    BCSCTL1&=~XT2OFF;

    do{
        IFG1&=~OFIFG;
        for(i=0xff;i>0;i++);
    }while(IFG1&OFIFG);

    BCSCTL2|=SELM_2;
}
void delay_1us(void)
{
    unsigned int i;
    for(i=0;i<8;i++);
}
void delay250ms(void)
{
    unsigned int i,j,k;
    for(i=0;i<250;i++)
        for(j=0;j<1000;j++)
            for(k=0;k<8;k++);
}
void delay1s(void)
{
    unsigned int i,j,k;
    for(i=0;i<1000;i++)
        for(j=0;j<1000;j++)
            for(k=0;k<8;k++);
}
void delay40ms(void)
{
```

```
    unsigned int i,j,k;
    for(i=0;i<40;i++)
        for(j=0;j<1000;j++)
            for(k=0;k<8;k++);
}
```

2、Nokai5110.h:

```
//File: Nokia5110
//Date: 2012/2/12
//Time: 10:25
//Note: Nokia 5210 LCD 液晶显示
//每 8 位显示数据是低位在上，高位在下
//测试硬件
//单片机:    MSP430F149


//防止重复引用
#ifndef __LCD5110_V1_H__
#define __LCD5110_V1_H__


//指令宏定义
#define X_Col_Addr        0x80    //定位到第 0 列指令(列起始地址)(0 - 83)
#define Y_Page_Addr       0x40    //定位到第 0 页指令(页起始地址)(0 - 5)


//功能宏定义
//液晶复位
#define LCD_reset_hard RST_L;RST_H //硬件复位
#define LCD_reset_soft LCD_reset_5110()     //软件复位
//液晶显示控制(不影响 DDRAM)
#define LCD_show_blank    LCD_write_cmd(0x08)    //显示空白
#define LCD_show_normal    LCD_write_cmd(0x0c)    //正常显示
#define LCD_show_black    LCD_write_cmd(0x09)    //显示全黑
#define LCD_show_inverse    LCD_write_cmd(0x0d)    //反色显示
//便于理解
#define LCD_write_cmd(cmd) LCD_write_byte(cmd, 0) //写入指令
#define LCD_write_dat(dat) LCD_write_byte(dat, 1) //写入数据


//数据接口定义
#define CLK_H      P2OUT|=BIT0        //串行时钟                //上升沿写入数据
#define CLK_L      P2OUT&=~BIT0
```

```
#define DIN_H     P2OUT|=BIT1        //串行数据输入          //先高后低
#define DIN_L     P2OUT&=~BIT1
#define DC_H      P2OUT|=BIT2        //数据指令控制端        //高电平数据，低电平指令
#define DC_L      P2OUT&=~BIT2
#define CE_H      P2OUT|=BIT3        //片选使能              //低电平有效
#define CE_L      P2OUT&=~BIT3
#define RST_H     P2OUT|=BIT4        //LCD 复位端            //低电平复位
#define RST_L     P2OUT&=~BIT4


//函数声明(私有)
void LCD_write_byte(unsigned char wbyte, unsigned char dat_cmd);//写入字节
void LCD_pos_char(unsigned char x, unsigned char y);     //液晶定位(8*8)
void LCD_reset_5110(void);                //复位 LCD5510


//函数声明(公有)

//清屏参数(清 DDRAM)
void LCD_clr_scr(void);


//液晶字节定位(1*1)
//液晶规划:
//x: 0 - 83
//y: 0 - 5
void LCD_pos_byte(unsigned char x, unsigned char y);


//液晶字符输出(8*16 字体)
//x: 0 - 9
//y: 0 - 2
void LCD_printc_8x16(unsigned char x, unsigned char y, unsigned char c_dat);


//液晶字符串输出(8*16 字体)
//x: 0 - 9
//y: 0 - 2
void LCD_prints_8x16(unsigned char x, unsigned char y, const unsigned char    *s_dat);


//初始化 LCD5510
void LCD5510_Init(void);
```

```
//液晶字符输出(6*8 字体)
//x: 0 - 13
//y: 0 - 5
void LCD_printc_6x8(unsigned char x, unsigned char y, unsigned char c_dat);


//液晶字符串输出(6*8 字体)
//x: 0 - 13
//y: 0 - 5
void LCD_prints_6x8(unsigned char x, unsigned char y, unsigned char *s_dat);


//液晶字符串输出,自动换行(6*8 字体)
//x: 0 - 13
//y: 0 - 5
void LCD_printsl_6x8(unsigned char x, unsigned char y, unsigned char *s_dat);


//液晶汉字输出(16*16 字体)
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//x: 0 - 4
//y: 0 - 2
void LCD_printch_16x16(unsigned char x, unsigned char y, unsigned char    *h_dat);


//显示 84X48 图片
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
void LCD_picture_84x48(unsigned char    *img_dat);


//定位显示指定大小图片
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//pag: 0 - 5      页坐标
//col:    0 - 83      列坐标
//x: 0 - (83-col)  图片宽
//y: 0 - (47-pag*8)  图片高
void LCD_pos_picture(unsigned char col, unsigned char pag, unsigned char x, unsigned char y,
unsigned char *img_dat);


//定位输出数字
//x: 0 - 13
//y: 0 - 5
```

```
//num: 0 - 65535  要显示的数字
//num_bit: 0 - 5  数字的位数
void LCD_printn_6x8(unsigned char x, unsigned char y, unsigned int num, unsigned char
num_bit);


//包含文件
#include "Nokia5110.C"


#endif
```

3：Nokia5110.c:

```
//写入一个字节(数据或指令)
//wbyte:  待写入的数据
//dat_cmd: 1-数据, 0-指令
void LCD_write_byte(unsigned char wbyte, unsigned char dat_cmd)
{
 unsigned char i;
 CE_L;     //使能
 if(dat_cmd ==1)         DC_H;//数据
 else                    DC_L;//指令
 for(i = 8; i; i--) //8 位数据,  先高后低
 {
  if(wbyte & 0x80) {DIN_H;}
  else       {DIN_L;}
  CLK_L;
  wbyte <<= 1; //移位(延时)
  CLK_H; //上升沿写入
 }
 CE_H;     //禁止
}


//写入 n 个字节(数据)(ROM)
/
//显示清屏(清 DDRAM)
void LCD_clr_scr(void)
{
 unsigned int i;
 LCD_write_cmd(X_Col_Addr);
 LCD_write_cmd(Y_Page_Addr);
```

```
  for(i = 504; i; i--) LCD_write_dat(0x00);
}


//显示清行
//num: 0 - 5
void LCD_clr_row(unsigned char num)
{
  unsigned char i;
  LCD_pos_byte(0, num);
  for(i = 84; i; i--) LCD_write_dat(0x00);
}


//液晶字节定位(1*1)
//液晶规划:
//x: 0 - 83
//y: 0 - 5
void LCD_pos_byte(unsigned char x, unsigned char y)
{
  x |= X_Col_Addr;
  y |= Y_Page_Addr;
  LCD_write_cmd(x); //列地址
  LCD_write_cmd(y); //页地址
}


//液晶字符输出(8*16 字体)
//x: 0 - 9
//y: 0 - 2
void LCD_printc_8x16(unsigned char x, unsigned char y, unsigned char c_dat)
{
  unsigned char i, j;
  c_dat -= 32;
  x <<= 3;    //8
  y <<= 1;    //16
  for(j = 0; j < 2; j++)
  {
    LCD_pos_byte(x, (y + j));
    for(i = 0; i < 8; i++)
    LCD_write_dat(Font_code_8x16[c_dat][8 * j + i]);
  }
}
```

```
//液晶字符串输出(8*16 字体)
//x: 0 - 9
//y: 0 - 2
void LCD_prints_8x16(unsigned char x, unsigned char y, const unsigned char    *s_dat)
{
 while(*s_dat && x < 10)
 {LCD_printc_8x16(x++, y, *s_dat); s_dat++;}
}


//复位 LCD5110
void LCD_reset_5110(void)
{
 LCD_clr_scr();       //清全屏
 LCD_write_cmd(0x25);   //省电模式，水平寻址，扩展指令
 LCD_write_cmd(0x04);   //VLCD 温度系数 0
 LCD_write_cmd(0x10);   //设置偏置系统(BSx)
 LCD_write_cmd(0xc0);   //设置电压 VLCD = 3.06 + 0.06*Vop;
 LCD_write_cmd(0x24);   //省电模式，水平寻址，常规指令
 LCD_write_cmd(0x08);   //显示空白
 LCD_write_cmd(Y_Page_Addr); //起始页地址 0
 LCD_write_cmd(X_Col_Addr); //起始列地址 0
}


//初始化 LCD5110
void LCD5110_Init(void)
{
 LCD_reset_hard;       //硬件复位
// LCD_reset_soft;       //软件复位
 LCD_write_cmd(0x21);   //工作模式，水平寻址，扩展指令
 LCD_write_cmd(0x06);   //VLCD 温度系数 2
 LCD_write_cmd(0x13);   //设置偏置系统(BSx) 1:48
 LCD_write_cmd(0xc8);   //设置电压 VLCD = 3.06 + 0.06*Vop，对比度调整
 LCD_write_cmd(0x20);   //工作模式，水平寻址，常规指令
 LCD_write_cmd(0x0c);   //普通模式
 LCD_write_cmd(Y_Page_Addr); //起始页地址 0
 LCD_write_cmd(X_Col_Addr); //起始列地址 0
 LCD_clr_scr();       //清全屏
}

//液晶字符输出(6*8 字体)
//x: 0 - 13
```

```
//y: 0 - 5
void LCD_printc_6x8(unsigned char x, unsigned char y, unsigned char c_dat)
{
 unsigned char i;
 c_dat -= 32;   //查表
 x *= 6;       //宽 6
 LCD_pos_byte(x, y); //坐标
 for(i = 0; i < 6; i++) LCD_write_dat(Font_code_6x8[c_dat][i]);
}


//液晶字符串输出(6*8 字体)
//x: 0 - 13
//y: 0 - 5
void LCD_prints_6x8(unsigned char x, unsigned char y, unsigned char *s_dat)
{
 while(*s_dat && x < 14) {LCD_printc_6x8(x++, y, *s_dat); s_dat++;}
}


//液晶字符串输出,自动换行(6*8 字体)
//x: 0 - 13
//y: 0 - 5
void LCD_printsl_6x8(unsigned char x, unsigned char y, unsigned char *s_dat)
{
 while(*s_dat)
 {
  LCD_printc_6x8(x++, y, *s_dat);
  s_dat++;
  if(x == 14) {x = 0; y++;}
  if(y == 6)    {y = 0;}
 }
}


//液晶汉字输出(16*16 字体)
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//x: 0 - 4
//y: 0 - 2
void LCD_printch_16x16(unsigned char x, unsigned char y, unsigned char    *h_dat)
{
 unsigned char i, j;
 x <<= 4;   //字宽 16
 y <<= 1; //字高 16
```

```
  for(j = 0; j < 2; j++)
  {
   LCD_pos_byte(x, (y + j)); //坐标
   for(i = 0; i < 16; i++) LCD_write_dat(h_dat[16 * j + i]);
  }
}


//显示 84X48 图片
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
void LCD_picture_84x48(unsigned char    *img_dat)
{
  unsigned int i;
  for(i = 0; i < 504; i++) LCD_write_dat(img_dat[i]);
}


//定位显示指定大小图片
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//pag: 0 - 5      页坐标
//col:    0 - 83      列坐标
//x: 0 - (83-col)  图片宽
//y: 0 - (47-pag*8)  图片高
void LCD_pos_picture(unsigned char col, unsigned char pag, unsigned char x, unsigned char y,
unsigned char *img_dat)
{
  unsigned char i, j;
  y = (y + 4) >> 3;     //四舍五入
  for(j = 0; j < y; j++)
  {
   for(i = 0; i < x; i++)
   {
    LCD_pos_byte(col + i, pag + j);    //坐标
    LCD_write_dat(img_dat[j * x + i]); //数据
   }
  }
}

//定位输出数字
//x: 0 - 13
//y: 0 - 5
//num: 0 - 65535  要显示的数字
//num_bit: 0 - 5  数字的位数
void LCD_printn_6x8(unsigned char x, unsigned char y, unsigned int num, unsigned char
```

```
num_bit)
{
  signed      char i;
  unsigned char ii;
  unsigned char dat[6];

  for(i = 0; i < 6; i++) dat[i] = 0;    //初始化数据
  i = 0;
  while(num / 10)            //拆位
  {
    dat[i] = num % 10;       //最低位
    num /= 10; i++;
  }
  dat[i] = num;              //最高位
  ii = i;                    //保存 dat 的位数
  for(; i >= 0; i--) dat[i] += 48; //转化成 ASCII
  for(i = 0; i < num_bit; i++)
  LCD_printc_6x8(x + i, y, ' ');     //清显示区域
  for(i = ii; i >= 0; i--)
  LCD_printc_6x8(x++, y, dat[i]);     //输出数值
}
```

4、Font_code(8x16).c:

```
//字库码的引索



//      字体: Fixedsys
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//查表方法: 要显示的 ASCII 码-32 就可以得到字库码的指针
const unsigned char Font_code_8x16[][16] = {
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},//    0
{0x00,0x00,0x70,0xF8,0xF8,0x70,0x00,0x00,0x00,0x00,0x00,0x0D,0x0D,0x00,0x00,0x00},//! 1
{0x00,0x38,0x38,0x00,0x00,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},//" 2
{0x00,0x20,0xF8,0xF8,0x20,0xF8,0xF8,0x20,0x00,0x02,0x0F,0x0F,0x02,0x0F,0x0F,0x02},//# 3
{0x00,0x30,0x78,0xCE,0x8E,0x18,0x10,0x00,0x00,0x04,0x0C,0x38,0x39,0x0F,0x06,0x00},//$ 4
{0x18,0x3C,0x24,0xBC,0xD8,0x60,0x30,0x00,0x00,0x06,0x03,0x0D,0x1E,0x12,0x1E,0x0C},//
% 5
{0x00,0xB0,0xF8,0x48,0x78,0x30,0x00,0x00,0x00,0x07,0x0F,0x08,0x09,0x07,0x0F,0x09},//& 6
{0x00,0x00,0x00,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},//' 7
{0x00,0x00,0xC0,0xF0,0x38,0x08,0x00,0x00,0x00,0x00,0x07,0x1F,0x38,0x20,0x00,0x00},//( 8
{0x00,0x00,0x08,0x38,0xF0,0xC0,0x00,0x00,0x00,0x00,0x20,0x38,0x1F,0x07,0x00,0x00},//) 9
{0x00,0x80,0xA0,0xE0,0xC0,0xE0,0xA0,0x80,0x00,0x00,0x02,0x03,0x01,0x03,0x02,0x00},/
```

//字库码的引索

//　　字体: LCD1602 字体
//取码规则: 低位在前, 列行扫描, 阴码(1-亮, 0-灭)
//查表方法: 要显示的 ASCII 码-32 就可以得到字库码的指针
const unsigned char Font_code_6x8[][6] = {
{0x00,0x00,0x00,0x00,0x00,0x00},// (0)
{0x00,0x00,0x00,0x4F,0x00,0x00},//!(1)
{0x00,0x00,0x07,0x00,0x07,0x00},//"(2)
{0x00,0x14,0x7F,0x14,0x7F,0x14},//#(3)
{0x00,0x24,0x2A,0x7F,0x2A,0x12},//$(4)
{0x00,0x23,0x13,0x08,0x64,0x62},//%(5)
{0x00,0x36,0x49,0x55,0x22,0x50},//&(6)
{0x00,0x00,0x05,0x03,0x00,0x00},//'(7)
{0x00,0x00,0x1C,0x22,0x41,0x00},//((8)
{0x00,0x00,0x41,0x22,0x1C,0x00},//)(9)
{0x00,0x14,0x08,0x3E,0x08,0x14},/
6、picture.c:
unsigned char pic[] =
{//84*48
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0xC0,0x70,0x20,0x30,0x20,0x40,0x80,0xC0,0x40,0x20,0x30,0x20,0x20,
0x20,0x20,0x50,0xC0,0x80,0x20,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x80,0xC0,0x40,0x40,0x80,0xC0,0x40,0xA0,0x60,0xB0,0xB0,0xD0,0xF0,0x70,0xF0,
0xB0,0x60,0xA0,0x60,0x00,0x00,0x00,0xC0,0x60,0x30,0x18,0x0C,0x04,0x06,0x82,0x46,
0x02,0xE2,0x02,0x02,0x02,0x06,0x06,0x8F,0x8C,0x8F,0x9F,0xB0,0x30,0x74,0x63,0xC0,
0x8E,0x0F,0x0E,0x00,0x00,0x00,0x80,0x80,0xE3,0x1C,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x40,0xB8,0x44,0xE2,0x43,0x03,0x06,0xF8,0x45,0xF2,0x1C,0xC7,
0xE2,0x02,0x03,0x02,0x0D,0x3A,0xC5,0x5A,0x00,0x00,0x3F,0xF7,0xC0,0x00,0x00,0x00,
0x3C,0xC2,0x3D,0xC2,0x15,0x40,0x00,0x00,0x00,0x00,0x02,0x03,0x05,0x05,0x07,0x01,
0x06,0x01,0x00,0x00,0x00,0x01,0x03,0x03,0x05,0x0B,0x21,0x9E,0x21,0xDF,0x2C,0xF8,
0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0xC0,0x20,0xD0,0xF8,0x90,0x9C,0x97,0x92,0xF4,0xF4,0x0E,0xA2,0x55,
0xAA,0x03,0x5E,0xA8,0x11,0x50,0xB0,0x58,0xAC,0x07,0x79,0x86,0x00,0x00,0x00,0x01,
0x03,0x07,0x0E,0xAC,0xFC,0x59,0xBA,0xF1,0xB6,0x35,0x32,0x34,0xE8,0x30,0x60,0x20,
0x20,0x20,0x30,0xE0,0x30,0x10,0x10,0x18,0x88,0x6C,0x1E,0x03,0x00,0x00,0x01,0xA2,
0x14,0xEA,0x15,0xEA,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xFD,0xA2,0x5D,0xA2,0xFF,0x95,0x6B,0x95,0x69,
0x86,0x58,0xA2,0x55,0xAA,0x40,0xB5,0x4A,0xA0,0x55,0xAA,0xD4,0xA9,0x56,0xA9,0x56,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x0E,0x0D,0x3A,0x35,0x6B,0xD6,0xEC,

0x9D,0xEE,0x18,0xE8,0x18,0xE0,0x1C,0xE7,0x14,0xEA,0x96,0xE1,0xD1,0xA8,0xE0,0xE8,
0xF4,0x2A,0x39,0x1A,0x1D,0x0E,0x07,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x03,0x07,0x06,
0x3F,0x7D,0xFE,0xF9,0xBE,0x39,0x1E,0xBD,0xFC,0x7F,0x3C,0x77,0xCE,0x77,0x8B,0x75,
0x8A,0x75,0x8A,0xF5,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x01,0x01,0x01,0x03,0x0B,0x0F,0x03,0x0D,0x03,0x0C,0x03,0x0C,
0x03,0x0C,0x03,0x0D,0x03,0x0F,0x0E,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x01,0x01,0x00,0x00,0x00,0x00,0x08,
0x0F,0x05,0x0A,0x05,0x0A,0x05,0x0A,0x07};
unsigned char pic1[] = {//72*14
0x1C,0x08,0xFE,0x08,0x08,0xFA,0xAF,0xFA,0x08,0x00,0x00,0x00,0x80,0x7C,0x80,0x00,
0x00,0x00,0x94,0x56,0x3C,0xD4,0x00,0x38,0x28,0x38,0x00,0x00,0xF4,0x94,0x94,0xDC,
0x00,0x00,0x00,0x00,0xC1,0xFF,0xFC,0x8D,0x3D,0x78,0x00,0x00,0x00,0x00,0x01,0x01,
0x09,0x05,0x1D,0x3D,0x7D,0x7F,0x1B,0x0B,0x83,0xFF,0x7F,0x3F,0x8F,0x87,0x03,0x83,
0xFF,0xFF,0xFF,0xFF,0xFF,0x0F,0x00,0x00,0x28,0x28,0x28,0x28,0x28,0x29,0x28,0x2B,
0x28,0x28,0x29,0x29,0x28,0x28,0x28,0x29,0x28,0x28,0x28,0x28,0x20,0x04,0x0A,0x12,
0x24,0x12,0x0A,0x24,0x20,0x28,0x28,0x00,0x10,0x18,0x3C,0x3F,0x3F,0x3F,0x3F,0x3F,
0x38,0x20,0x00,0x03,0x30,0x30,0x20,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x02,
0x21,0x30,0x38,0x3F,0x3C,0x3C,0x3F,0x3F,0x3F,0x3F,0x3F,0x0F,0x21,0x28,0x28,0x28};
unsigned char pic2[] = {//72*28
0xFF,0x4D,0x43,0x41,0x41,0x41,0x41,0x41,0x01,0x01,0x03,0x05,0x0B,0x17,0x1F,0x1D,
0x13,0x15,0xBB,0xF5,0xEB,0xD5,0xAB,0xF5,0xEB,0xD5,0xAB,0xD5,0xAB,0xD5,0xAB,0x55,
0x2B,0x77,0xBF,0x00,0x14,0x14,0x94,0xFC,0x92,0x50,0xFE,0x10,0x92,0x54,0x10,0x00,
0x00,0xE0,0x00,0xF8,0x00,0x02,0x1C,0x00,0x00,0x60,0x80,0x00,0x00,0x70,0xF0,0xF0,
0xE0,0xC0,0xE0,0xF0,0xF0,0x70,0x00,0x00,0xFF,0x46,0x0F,0xDF,0xAF,0x47,0xAF,0x1E,
0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x0E,0x0C,0xEE,0x46,0x86,0x56,
0xBE,0x5C,0x38,0x00,0x06,0x09,0xC8,0x18,0xEC,0x55,0xAA,0x00,0x01,0x89,0x88,0x8F,
0x84,0x84,0x82,0x83,0x84,0x88,0x0E,0x00,0x01,0x80,0x80,0x8F,0x88,0x88,0x88,0x88,
0x8F,0x80,0x01,0x00,0x00,0x00,0x00,0x81,0x03,0x07,0x83,0x01,0x00,0x00,0x00,0x00,
0xFF,0x55,0xBE,0xE0,0x81,0x01,0x01,0x01,0x00,0x00,0x06,0x08,0x40,0x40,0x40,0x40,
0x40,0x00,0x00,0x00,0x01,0x03,0x03,0x03,0x02,0x81,0x40,0x20,0x10,0xFC,0xA9,0x58,
0xAB,0x55,0xAA,0x00,0x00,0xFF,0x02,0xBA,0xAA,0xAA,0xFE,0xAA,0xAA,0xBB,0x80,0x00,
0x08,0x08,0x08,0x08,0x08,0xFF,0x08,0x08,0x08,0x08,0x08,0x00,0x08,0x04,0xFE,0x89,
0x44,0x33,0x02,0xFA,0x02,0x32,0xC6,0x00,0x0F,0x05,0x0A,0x0F,0x0A,0x05,0x0A,0x0C,
0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x08,0x04,0x04,0x02,0x02,
0x01,0x00,0x00,0x00,0x00,0x0F,0x0A,0x05,0x0A,0x05,0x0A,0x00,0x03,0x00,0x00,0x03,
0x00,0x02,0x03,0x02,0x00,0x02,0x03,0x00,0x00,0x00,0x00,0x02,0x02,0x03,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x02,0x02,0x03,0x00,0x00,0x00,0x00};

MSP430F149 学习之 12864 显示

```
#include "io430.h"
#include    "in430.h"
```

```
#define SET_DATA    P2OUT|=BIT0
#define SET_INC     P2OUT&=~BIT0
#define SET_READ    P2OUT|=BIT1
#define SET_WRITE P2OUT&=~BIT1
#define SET_EN      P2OUT|=BIT2
#define CLR_EN      P2OUT&=~BIT2
#define io_LCD12864_DATAPORT    P4OUT
void v_Lcd12864CheckBusy_f(void)
{
   unsigned int nTimeOut =0;
   SET_INC;
   SET_READ;
   CLR_EN;
   SET_EN;
   P4DIR=0X00;
   while((P4IN&0X80)&&(++nTimeOut!=0));
   CLR_EN;
   SET_INC;
   SET_READ;
}
void v_Lcd12864SendCmd_f(unsigned char byCmd)
{
   v_Lcd12864CheckBusy_f();
   SET_INC;
   SET_WRITE;
   CLR_EN;
   P4DIR=0XFF;
   io_LCD12864_DATAPORT=byCmd;
   _NOP();
   _NOP();
   SET_EN;
   _NOP();
   _NOP();
   CLR_EN;
   SET_READ;
   SET_INC;
}
void v_Lcd12864SendData_f(unsigned char byData)
{
   v_Lcd12864CheckBusy_f();
   SET_DATA;
   SET_WRITE;
   CLR_EN;
   P4DIR=0XFF;
```

```
    io_LCD12864_DATAPORT=byData;
    _NOP();
    _NOP();
    SET_EN;
    _NOP();
    _NOP();
    CLR_EN;
    SET_READ;
    SET_INC;
}
void v_DelayMs_f(unsigned int nDelay)
{
    unsigned int i;
    for(;nDelay>0;nDelay--) for(i=125;i>0;i--);
}
void v_Lcd12864Init_f(void)
{
    v_Lcd12864SendCmd_f(0x30);
    v_DelayMs_f(50);
    v_Lcd12864SendCmd_f(0x01);
    v_DelayMs_f(50);
    v_Lcd12864SendCmd_f(0x06);
    v_DelayMs_f(50);
    v_Lcd12864SendCmd_f(0x0c);
}
void v_Lcd12864SetAddress_f(unsigned char x,unsigned char y)
{
    unsigned char byAddress;
    switch(y)
    {
    case 0: byAddress=0x80+x;break;
    case 1: byAddress=0x90+x;break;
    case 2: byAddress=0x88+x;break;
    case 3: byAddress=0x98+x;break;
    default:break;
    }
    v_Lcd12864SendCmd_f(byAddress);
}
void v_Lcd12864PutString_f(unsigned char x, unsigned char y, unsigned char *pData )
{
    v_Lcd12864SetAddress_f(x,y);
    while(*pData!='\0')
    {
        v_Lcd12864SendData_f(*pData++);
```

```
    }
}
int main( void )
{
    // Stop watchdog timer to prevent time out reset
    WDTCTL = WDTPW + WDTHOLD;
    P2DIR=0XFF;
    v_Lcd12864Init_f();
    v_Lcd12864PutString_f(1,0,"天狼 skywolf");
    v_Lcd12864PutString_f(0,1,"*LCD12864ST7920*") ;
    v_Lcd12864PutString_f(2,2, "新浪微博") ;
    v_Lcd12864PutString_f(1,3, "天狼正在行动") ;
    while(1);

    //return 0;
}
```