一、基础实验【**10**个】

1、入门试验：LED闪烁（1个）
2、时钟实验：设置MCLK、ACLK、SMCLK（1个）
3、低功耗实验：设置低功耗模式（1个）
4、IO端口试验： IO端口寄存器设置（1个）
5、定时器：看门狗定时器、TimerA寄存器设置（2个）
6、比较器：比较器A寄存器（1个）
7、Flash：flash读写（1个）
8、异步通信：异步通信寄存器设置（1个）
9、ADC：ADC12寄存器设置（1个）

二、开发板模块简单程序【**56**个】

1、LED流水灯实验（红、黄、绿）

（1）LED1：检测开发板
（2）LED2：普通IO控制闪烁
（3）LED3：PWM信号控制闪烁

2、蜂鸣器实验

（1）蜂鸣器1：单频音（步进变音调）
（2）蜂鸣器2：奏乐（祝你平安）

3、数码管实验

（1）数码管1（显示123456）
（2）数码管2（动态显示0~F）
（3）数码管3（流动光圈）
（4）数码管4（来回光标）

4、4×1独立按键实验

（1）4×1键盘1：扫描数码管显示
（2）4×1键盘2：中断数码管显示
（3）4×1键盘3：控制LED
（4）4×1键盘4：控制蜂鸣器

5、4×4矩阵键盘实验

（1）4×4键盘1：行列扫描数码管显示
（2）4×4键盘2：行列扫描1602液晶显示

（3）4×4键盘3：控制LED蜂鸣器

6、1602液晶实验

（1）1602液晶1：动态字符显示
（2）1602液晶2：静态字符显示
（3）1602液晶3：内部时钟显示

7、3.3V-5V电平转换实验

（1）电平转换1：输出5V方波
（2）电平转换2：输出不同占空比的方波
（3）电平转换3：MCLK，SMCLK，ACLK

8、RS232接口实验

（1）RS232接口1：MCU发送数据PC机显示
（2）RS232接口2：按键控制MCU发送数据PC机显示
（3）RS232接口3：PC机发送数据MCU液晶显示
（4）RS232接口4：MCU回发接收到的PC机数据
（5）RS232接口5：RS232控制蜂鸣器

9、RS485接口实验

（1）RS485接口1：发送程序
（2）RS485接口2：接收程序

10、USB接口实验

（1）USB接口1：简单连接测试
（2）USB接口2：USB接收数据
（3）USB接口3：USB发送数据

11、PS2接口实验

（1）PS2接口1： PS2控制1602显示
（2）PS2接口2： PS2控制数码管显示
（3）PS2接口3：PS2控制LED和蜂鸣器

12、12-Bit高精度温度传感器实验

（1）温度传感器1：DS18B20在数码管显示
（2）温度传感器2：DS18B20在液晶显示

13、RTC实时时钟实验

（1）实时时钟1：DS1302测试

（2）实时时钟2：DS1302电子钟

14、2k Bit EEPROM实验

（1）EEPROM1：AT24C02测试
（2）EEPROM2：读出数据通过串口在PC机显示

15、12-Bit模数转换器（ADC）接口实验

（1）模数转换器1：ADC在数码管显示
（2）模数转换器2：ADC在1602液晶在显示
（3）模数转换器3：ADC通过串口在PC机显示

16、8-Bit数模转换器（DAC）实验

（1）数模转换器1：DAC控制LED
（2）数模转换器2：DAC输出电压，ADC采样转换并在液晶上显示

17、12864液晶实验（与12864液晶配套）

（1）12864液晶并口1：字符显示
（2）12864液晶并口2：汉字显示
（3）12864液晶并口3：图形显示
（4）12864液晶并口4：综合演示
（5）12864液晶串口5：字符显示
（6）12864液晶串口6：汉字显示
（7）12864液晶串口7：图形显示
（8）12864液晶串口8：综合演示

18、射频模块CC1000实验

（1）射频模块1：发送数据
（2）射频模块2：接收数据

19、ucos移植

注：17、18程序随模块赠送

三、开发板综合程序【30】

1、键盘综合实验

（1）4×4键盘+蜂鸣器+LED+数码管显示
（2）4×4键盘+蜂鸣器+LED+1602液晶显示
（3）4×4键盘+蜂鸣器+LED+PC机显示
（4）PS2键盘+UART+PC机显示

（5）PS2键盘+USB+PC机显示

2、接口综合实验

（1）USB转UART
（2）UART转USB
（3）RS232转RS485
（4）RS485转RS232

3、温度时间综合实验

（1）DS18B20 + DS1302 + 数码管
（2）DS18B20 + DS1302 + USB
（3）DS18B20 + DS1302 + UART
（4）DS18B20 + DS1302 + 1602

4、AD DA综合实验

（1）ADC + 1602
（2）ADC + UART
（3）ADC + USB
（4）DAC + LED + KEY
（5）DAC + UART
（6）DAC + USB
（7）ADC + UART + DS1302
（8）ADC + DAC + 1602 + KEY
（9）ADC + DAC + UART + KEY

5、其他综合实验

（1）AT24C02高级应用（搜索，擦除，读出全部）
（2）DS1302高级应用（内部RAM存取数据）

6、12864液晶综合实验

（1）汉字库
（2）图形库

7、3.2寸TFT触摸屏实验

（1）静态图片
（2）动画

```
/*****************************************************
程序功能：BoardConfig.h     头文件
-----------------------------------------------
*****************************************************/
typedef unsigned char uchar;
typedef unsigned int    uint;

//控制位的宏定义
#define Ctrl_Out    P3DIR |= BIT3 + BIT6 + BIT7;
#define Ctrl_0      P3OUT &= ~(BIT3 + BIT6 + BIT7)
#define SRCLK_1     P3OUT |= BIT7
#define SRCLK_0     P3OUT &= ~BIT7
#define SER_1       P3OUT |= BIT6
#define SER_0       P3OUT &= ~BIT6
#define RCLK_1      P3OUT |= BIT3
#define RCLK_0      P3OUT &= ~BIT3

//板上资源配置函数
void BoardConfig(uchar cmd)
{
    uchar i;

    Ctrl_Out;
    Ctrl_0;

    for(i = 0; i < 8; i++)
    {
        SRCLK_0;
        if(cmd & 0x80)    SER_1;
        else                SER_0;
        SRCLK_1;
        cmd <<= 1;
    }
    RCLK_1;
    _NOP();
    RCLK_0;
}
```

```
/*****************************************************
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>
#include   "BoardConfig.h"
```

```
/****************主函数****************/
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;          //关闭看门狗
    BoardConfig(0xf0);                 //关闭数码管和电平转换，打开流水灯

    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
}
```

```
/*********************************************
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>
#include   "BoardConfig.h"

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
```

```
    BoardConfig(0xf0);

    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;

    _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;
        }
}
```

```
/************************************************************
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>
#include   "BoardConfig.h"
void main(void)
```

```
{
  WDTCTL = WDTPW + WDTHOLD;                    // 关狗
  BoardConfig(0xb0);                          // 关闭数码管和电平转换，打开流水灯
  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
  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
}
```

```
//*******************************************************************************************
//  MSP-FET430P140 Demo - Basic Clock, Output Buffered SMCLK, ACLK and MCLK
//
//  Description: Output buffered MCLK, SMCLK and ACLK.
//  ACLK = LFXT1 = 32768, MCLK = DCO Max, SMCLK = XT2
//  //* XTAL's REQUIRED - NOT INSTALLED ON FET *//
//  //* Min Vcc required varies with MCLK frequency - refer to datasheet *//
//
//                      MSP430F149
//                  -----------------
//            /|\|                  XIN|-
//             | |                     | 32k
//             --|RST           XOUT|-
//               |                     |
//               |                 XT2IN|-
//               |                     | XTAL (455k - 8Mhz)
//               |RST            XT2OUT|-
//               |                     |
//               |                P5.4|-->MCLK = DCO Max
//               |                P5.5|-->SMCLK = XT2
//               |                P5.6|-->ACLK = 32kHz
//
//  M. Buccini
//  Texas Instruments Inc.
//  Feb 2005
//  Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************
```

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

void main(void)
{
   BoardConfig(0xb8);
   WDTCTL = WDTPW +WDTHOLD;                          // Stop Watchdog Timer
   DCOCTL = DCO0 + DCO1 + DCO2;                     // Max DCO
   BCSCTL1 = RSEL0 + RSEL1 + RSEL2;                 // XT2on, max RSEL
   BCSCTL2 |= SELS;                                 // SMCLK = XT2
   P5DIR |= 0x70;                                   // P5.6,5,4 outputs
   P5SEL |= 0x70;                                   // P5.6,5,5 options

   while(1)
   {
   }
}
```

```
//*******************************************************************************
//   MSP-FET430P140 Demo - Basic Clock, LPM3 Using WDT ISR, 32kHz ACLK
//
//   Description: This program operates MSP430 normally in LPM3, pulsing P3.4
//   at 4 second intervals. WDT ISR used to wake-up system. All I/O configured
//   as low outputs to eliminate floating inputs. Current consumption does
//   increase when LED is powered on P3.4. Demo for measuring LPM3 current.
//   ACLK= LFXT1/4= 32768/4, MCLK= SMCLK= default DCO
//   //* External watch crystal on XIN XOUT is required for ACLK *//
//
//
//                  MSP430F149
//               ---------------
//         /|\|                  XIN|-
//          | |                     | 32kHz
//          --|RST             XOUT|-
//            |                     |
//            |                 P3.5|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*******************************************************************************

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

```
void main(void)
{
   BoardConfig(0xb8);
   BCSCTL1 |= DIVA_2;                          // ACLK/4
   WDTCTL = WDT_ADLY_1000;                     // WDT 1s/4 interval timer
   IE1 |= WDTIE;                               // Enable WDT interrupt

   P1DIR = 0xFF;                               // All P1.x outputs
   P1OUT = 0;                                  // All P1.x reset
   P2DIR = 0xFF;                               // All P2.x outputs
   P2OUT = 0;                                  // All P2.x reset
   P3DIR = 0xFF;                               // All P3.x outputs
   P3OUT = 0x30;                               // All P3.x reset
   P4DIR = 0xFF;                               // All P4.x outputs
   P4OUT = 0;                                  // All P4.x reset
   P5DIR = 0xFF;                               // All P5.x outputs
   P5OUT = 0;                                  // All P5.x reset
   P6DIR = 0xFF;                               // All P6.x outputs
   P6OUT = 0x80;                               // All P6.x reset

   while(1)
   {
      uint i;
      _BIS_SR(LPM3_bits + GIE);                // Enter LPM3
      P3OUT &= ~BIT5;                          // Set P3.5 LED on
      for (i = 18000; i>0; i--);               // Delay
      P3OUT |= BIT5;                           // Clear P3.5 LED off
   }
}

#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer (void)
{
      _BIC_SR_IRQ(LPM3_bits);                  // Clear LPM3 bits from 0(SR)
}
```

```
//*******************************************************************************************
//   MSP-FET430P140 Demo - Software Toggle P3.4
//
//   Description: Toggle P3.4 by xor'ing P3.4 inside of a software loop.
//   ACLK= n/a, MCLK= SMCLK= default DCO ~800k
//
//                      MSP430F149
//                   -----------------
//               /|\|                 XIN|-
```

```
//               | |                         |
//               --|RST                XOUT|-
//                 |                         |
//                 |                   P3.4|-->LED
//
//  Dasheng
//  LiTian Electronic Inc.
//  Feb 2008
//  Built with IAR Embedded Workbench Version: 3.42A
//*******************************************************************************************

#include  <msp430x14x.h>
#include  "BoardConfig.h"

void main(void)
{
  BoardConfig(0xb8);
  WDTCTL = WDTPW + WDTHOLD;                         // Stop watchdog timer
  P3DIR |= BIT4;                                    // Set P3.4 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P3OUT ^= BIT4;                                  // Toggle P3.4 using exclusive-OR

    i = 50000;                                      // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
//*******************************************************************************************
//  MSP-FET430P140 Demo - WDT, Toggle P3.4, Interval Overflow ISR, DCO SMCLK
//
//  Description: Toggle P3.4 using software timed by the WDT ISR. Toggle rate
//  is approximately 30ms based on default ~ 800khz DCO/SMCLK clock source
//  used in this example for the WDT.
//  ACLK= n/a, MCLK= SMCLK= default DCO~ 800k
//
//              MSP430F149
//                  -----------------
//              /|\|                   XIN|-
//               | |                         |
//               --|RST                XOUT|-
```

```
//                  |                         |
//                  |                     P3.4|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*****************************************************************************
#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
  BoardConfig(0xbf);                              //关闭数码管、流水灯和电平转换
  WDTCTL = WDT_MDLY_32;                                 // Set Watchdog Timer interval to ~30ms
  IE1 |= WDTIE;                                  // Enable WDT interrupt
  P3DIR |= BIT4;                                 // Set P3.4 to output direction

  _BIS_SR(LPM0_bits + GIE);                      // Enter LPM0 w/ interrupt
}

// Watchdog Timer interrupt service routine
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
  P3OUT ^= BIT4;                                  // Toggle P3.4 using exclusive-OR
}
```

```
//*****************************************************************************
//   MSP-FET430P140 Demo - WDT, Toggle P3.4, Interval Overflow ISR, 32kHz ACLK
//
//   Description: Toggle P3.4 using software timed by WDT ISR. Toggle rate is
//   exactly 250ms based on 32kHz ACLK WDT clock source.    In this example the
//   WDT is configured to divide 32768 watch-crystal(2^15) by 2^13 with an ISR
//   triggered @ 4Hz.
//   ACLK= LFXT1= 32768, MCLK= SMCLK= DCO~ 800kHz
//   //* External watch crystal installed on XIN XOUT is required for ACLK *//
//
//              MSP430F149
//                   -----------------
//              /|\|                     XIN|-
//               | |                         | 32kHz
//               --|RST                XOUT|-
//                 |                         |
//                 |                     P3.4|-->LED
//
```

```
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*******************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
   BoardConfig(0xb8);
   WDTCTL = WDT_ADLY_250;                          // WDT 250ms, ACLK, interval timer
   IE1 |= WDTIE;                                   // Enable WDT interrupt
   P3DIR |= BIT4;                                  // Set P3.4 to output direction

   _BIS_SR(LPM3_bits + GIE);                       // Enter LPM3 w/interrupt
}

// Watchdog Timer interrupt service routine
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
   P3OUT ^= BIT4;                                  // Toggle P3.4 using exclusive-OR
}
```

```
//*******************************************************************************************
//   MSP-FET430P140 Demo - Timer_A, Toggle P3.4, CCR0 Cont. Mode ISR, DCO SMCLK
//
//   Description: Toggle P3.4 using software and TA_0 ISR. Toggles every
//   50000 SMCLK cycles. SMCLK provides clock source for TACLK.
//   During the TA_0 ISR, P3.4 is toggled and 50000 clock cycles are added to
//   CCR0. TA_0 ISR is triggered every 50000 cycles. CPU is normally off and
//   used only during TA_ISR.
//   ACLK = n/a, MCLK = SMCLK = TACLK = default DCO ~800kHz
//
//                MSP430F149
//             ---------------
//         /|\|                 XIN|-
//          | |                    |
//          --|RST            XOUT|-
//            |                    |
//            |                P3.4|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
```

```
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*******************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                       // Stop WDT
  BoardConfig(0xb8);                             //关闭数码管、流水灯和电平转换
  P3DIR |= BIT4;                                  // P3.4 output
  CCTL0 = CCIE;                                   // CCR0 interrupt enabled
  CCR0 = 50000;
  TACTL = TASSEL_2 + MC_2;                        // SMCLK, contmode

  _BIS_SR(LPM0_bits + GIE);                       // Enter LPM0 w/ interrupt
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P3OUT ^= BIT4;                                  // Toggle P3.4
  CCR0 += 50000;                                  // Add Offset to CCR0
}
```

```
//*******************************************************************************
//   MSP-FET430P140 Demo - Timer_A, Toggle P3.4, CCR0 Up Mode ISR, DCO SMCLK
//
//   Description: Toggle P3.4 using software and TA_0 ISR. Timer_A is
//   configured for up mode, thus the timer overflows when TAR counts
//   to CCR0. In this example, CCR0 is loaded with 20000.
//   ACLK = n/a, MCLK = SMCLK = TACLK = default DCO ~800kHz
//
//                MSP430F149
//             ---------------
//         /|\|                 XIN|-
//          | |                    |
//          --|RST            XOUT|-
//            |                    |
//            |                P3.4|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
```

```
//   Built with IAR Embedded Workbench Version: 3.42A
//*****************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                         // Stop WDT
  BoardConfig(0xb8);
  P3DIR |= BIT4;                                    // P3.4 output
  CCTL0 = CCIE;                                     // CCR0 interrupt enabled
  CCR0 = 20000;
  TACTL = TASSEL_2 + MC_1;                          // SMCLK, upmode

  _BIS_SR(LPM0_bits + GIE);                         // Enter LPM0 w/ interrupt
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P3OUT ^= BIT4;                                    // Toggle P3.4
}
```

```
//*****************************************************************************
//   MSP-FET430P140 Demo - Timer_A, Toggle P3.4, Overflow ISR, DCO SMCLK
//
//   Description: Toggle P3.4 using software and Timer_A overflow ISR.
//   In this example an ISR triggers when TA overflows. Inside the TA
//   overflow ISR P3.4 is toggled. Toggle rate is approximatlely 12Hz.
//   Proper use of the TAIV interrupt vector generator is demonstrated.
//   ACLK = n/a, MCLK = SMCLK = TACLK = default DCO ~800kHz
//
//                MSP430F149
//             ---------------
//        /|\|                 XIN|-
//         | |                    |
//         --|RST             XOUT|-
//           |                    |
//           |                P3.4|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*****************************************************************************
```

```
#include    <msp430x14x.h>
#include    "BoardConfig.h"
void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;                          // Stop WDT
   BoardConfig(0xb8);
   P3DIR |= BIT4;                                     // P3.4 output
   TACTL = TASSEL_2 + MC_2 + TAIE;                    // SMCLK, contmode, interrupt

   _BIS_SR(LPM0_bits + GIE);                          // Enter LPM0 w/ interrupt
}

// Timer_A3 Interrupt Vector (TAIV) handler
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void)
{
  switch( TAIV )
  {
     case    2: break;                               // CCR1 not used
     case    4: break;                               // CCR2 not used
     case 10: P3OUT ^= BIT4;                         // overflow
                 break;
  }
}
```

```
//*******************************************************************************
//   MSP-FET430P140 Demo - Timer_A, Toggle P3.4, Overflow ISR, 32kHz ACLK
//
//   Description: Toggle P3.4 using software and the Timer_A overflow ISR.
//   In this example an ISR triggers when TA overflows. Inside the ISR P3.4
//   is toggled. Toggle rate is exactly 0.5Hz. Proper use of the TAIV interrupt
//   vector generator is demonstrated.
//   ACLK = TACLK = 32768Hz, MCLK = SMCLK = default DCO ~800kHz
//   //* An external watch crystal on XIN XOUT is required for ACLK *//
//
//                 MSP430F149
//              ---------------
//          /|\|                 XIN|-
//           | |                    | 32kHz
//           --|RST            XOUT|-
//             |                    |
//             |                P3.4|-->LED
//
//   Dasheng
//   LiTian Electronic Inc.
```

```
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*******************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                         // Stop WDT
  BoardConfig(0xb8);
  P3DIR |= BIT4;                                    // P3.4 output
  TACTL = TASSEL_1 + MC_2 + TAIE;                   // ACLK, contmode, interrupt

  _BIS_SR(LPM3_bits + GIE);                         // Enter LPM3 w/ interrupt
}

// Timer_A3 Interrupt Vector (TAIV) handler
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void)
{
  switch( TAIV )
  {
    case   2:   break;                              // CCR1 not used
    case   4:   break;                              // CCR2 not used
    case 10:   P3OUT ^= BIT4;                       // overflow
                break;
  }
}
```

```
#include <msp430x14x.h>
#include "BoardConfig.h"
void main(void)
{
    WDTCTL = WDTHOLD + WDTPW;              //  关看门狗
    BoardConfig(0xb0);                     //开流水灯，关数码管和电平转换
    CACTL1 = CARSEL + CAREF0 + CAON ; //   Vcc/4 = - cmp
    CACTL2 = P2CA0;                        //  使用  CA0
    P2DIR = 0xff;
    P2OUT = 0xff;

    while(1)
    {
```

```
        if((CACTL2 | 0xfe) ==0xff)
        {                                       // 比较电压是否超过 0.25Vcc
          P2OUT &= ~BIT4;
          CACTL1 &= 0xfe;                       // CAIFG = 0
        }
        else
        {
          P2OUT |= BIT4;
        }
    }

}
```

```
//*******************************************************************************
//  MSP-FET430P140 Demo - Flash In-System Programming, Copy SegA to SegB
//
//  Description: This program first erases flash seg A, then it increments all
//  values in seg A, then it erases seg B, then  copies seg A to seg B.
//  Assumed MCLK 550kHz - 900kHz.
//  //* Set Breakpoint on NOP in the Mainloop to avoid Stressing Flash *//
//
//                      MSP430F149
//                  -----------------
//            /|\|                  XIN|-
//             | |                     |
//             --|RST                  XOUT|-
//               |                       |
//
//  M. Mitchell
//  Texas Instruments Inc.
//  Feb 2005
//  Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************

#include  <msp430x14x.h>
#include  "BoardConfig.h"

uchar  value;                                   // 8-bit value to write to segment A
uchar  DataBuffer[128];

// Function prototypes
void  write_SegA (uchar value);
void  copy_A2B (void);

void main(void)
```

```
{
  BoardConfig(0xb8);
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  FCTL2 = FWKEY + FSSEL0 + FN0;             // MCLK/2 for Flash Timing Generator
  value = 0;                                // Initialize value

  while(1)                                  // Repeat forever
  {
    write_SegA(value++);                    // Write segment A, increment value
    copy_A2B();                             // Copy segment A to B
    _NOP();                                 // SET BREAKPOINT HERE
  }
}

void write_SegA (uchar value)
{
  uchar *Flash_ptr;                         // Flash pointer
  uint i;

  Flash_ptr = (uchar *) 0x1080;             // Initialize Flash pointer
  FCTL1 = FWKEY + ERASE;                    // Set Erase bit
  FCTL3 = FWKEY;                            // Clear Lock bit
  *Flash_ptr = 0;                           // Dummy write to erase Flash segment

  FCTL1 = FWKEY + WRT;                      // Set WRT bit for write operation

  for (i=0; i<128; i++)
  {
    *Flash_ptr++ = value;                   // Write value to flash
  }

  FCTL1 = FWKEY;                            // Clear WRT bit
  FCTL3 = FWKEY + LOCK;                     // Set LOCK bit
}


void copy_A2B (void)
{
  uchar *Flash_ptrA;                        // Segment A pointer
  uchar *Flash_ptrB;                        // Segment B pointer
  uint i;

  Flash_ptrA = (uchar *) 0x1080;            // Initialize Flash segment A pointer
  Flash_ptrB = (uchar *) 0x1000;            // Initialize Flash segment B pointer
  FCTL1 = FWKEY + ERASE;                    // Set Erase bit
```

```
  FCTL3 = FWKEY;                            // Clear Lock bit
  *Flash_ptrB = 0;                          // Dummy write to erase Flash segment B
  FCTL1 = FWKEY + WRT;                      // Set WRT bit for write operation

  for (i=0; i<128; i++)
  {
    DataBuffer[i] = *Flash_ptrA++;
    *Flash_ptrB++ = DataBuffer[i];          // Copy value segment A to segment B
  }

  FCTL1 = FWKEY;                            // Clear WRT bit
  FCTL3 = FWKEY + LOCK;                     // Set LOCK bit
}
```

```
//*******************************************************************************************
//  MSP-FET430P140 Demo - USART0, Ultra-Low Pwr UART 2400 Echo ISR, 32kHz ACLK
//
//  Description: Echo a received character, RX ISR used. In the Mainloop UART0
//  is made ready to receive one character with interrupt active. The Mainloop
//  waits in LPM3. The UART0 ISR forces the Mainloop to exit LPM3 after
//  receiving one character which echo's back the received character.
//  ACLK = UCLK0 = LFXT1 = 32768, MCLK = SMCLK = DCO~ 800k
//  Baud rate divider with 32768hz XTAL @2400 = 32768Hz/2400 = 13.65 (000Dh)
//  //* An external watch crystal is required on XIN XOUT for ACLK *//
//
//                      MSP430F149
//                  -----------------
//            /|\|                  XIN|-
//             | |                     | 32kHz
//             --|RST              XOUT|-
//               |                     |
//               |                 P3.4|---------->
//               |                     | 2400 - 8N1
//               |                 P3.5|<----------
//
//
//  M. Buccini
//  Texas Instruments Inc.
//  Feb 2005
//  Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************

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

```
void main(void)
{
  BoardConfig(0xb8);
  WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
  P3SEL |= 0x30;                               // P3.4,5 = USART0 TXD/RXD
  ME1 |= UTXE0 + URXE0;                        // Enable USART0 TXD/RXD
  UCTL0 |= CHAR;                               // 8-bit character
  UTCTL0 |= SSEL0;                             // UCLK = ACLK
  UBR00 = 0x0D;                                // 32k/2400 - 13.65
  UBR10 = 0x00;                                //
  UMCTL0 = 0x6B;                               // Modulation
  UCTL0 &= ~SWRST;                             // Initialize USART state machine
  IE1 |= URXIE0;                               // Enable USART0 RX interrupt

// Mainloop
  for (;;)
  {
  _BIS_SR(LPM3_bits + GIE);                    // Enter LPM3 w/interrupt
  while (!(IFG1 & UTXIFG0));                   // USART0 TX buffer ready?
  TXBUF0 = RXBUF0;                             // RXBUF0 to TXBUF0
  }
}

// UART0 RX ISR will for exit from LPM3 in Mainloop
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
  _BIC_SR_IRQ(LPM3_bits);                      // Clear LPM3 bits from 0(SR)
}
```

```
//*******************************************************************************
//  MSP-FET430P140 Demo - USART0, Ultra-Low Pwr UART 9600 Echo ISR, 32kHz ACLK
//
//  Description: Echo a received character, RX ISR used. In the Mainloop UART0
//  is made ready to receive one character with interrupt active. The Mainloop
//  waits in LPM3. The UART0 ISR forces the Mainloop to exit LPM3 after
//  receiving one character which echo's back the received character.
//  ACLK = UCLK0 = LFXT1 = 32768, MCLK = SMCLK = DCO~ 800k
//  Baud rate divider with 32768hz XTAL @9600 = 32768Hz/9600 = 3.41 (0003h 4Ah )
//  //* An external watch crystal is required on XIN XOUT for ACLK *//
//
//                      MSP430F149
//                  -----------------
//            /|\|                  XIN|-
//             | |                     | 32kHz
//             --|RST                XOUT|-
```

```
//                |                     |
//                |                P3.4|----------->
//                |                     | 9600 - 8N1
//                |                P3.5|<-----------
//
//
//   M. Buccini
//   Texas Instruments Inc.
//   Feb 2005
//   Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"


void main(void)
{
   BoardConfig(0xb8);
   WDTCTL = WDTPW + WDTHOLD;                       // Stop WDT
   P3SEL |= 0x30;                                 // P3.4,5 = USART0 TXD/RXD
   ME1 |= UTXE0 + URXE0;                           // 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
   IE1 |= URXIE0;                                 // Enable USART0 RX interrupt

// Mainloop
   for (;;)
   {
   _BIS_SR(LPM3_bits + GIE);                      // Enter LPM3 w/interrupt
   while (!(IFG1 & UTXIFG0));                     // USART0 TX buffer ready?
   TXBUF0 = RXBUF0;                                // RXBUF0 to TXBUF0
   }
}

// UART0 RX ISR will for exit from LPM3 in Mainloop
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
   _BIC_SR_IRQ(LPM3_bits);                        // Clear LPM3 bits from 0(SR)
}
```

```
//*******************************************************************************************
//   MSP-FET430P140 Demo - USART0, UART 19200 Echo ISR, XT2 HF XTAL ACLK
//
//   Description: Echo a received character, RX ISR used. Normal mode is LPM0,
//   USART0 RX interrupt triggers TX Echo. Though not required, MCLK = XT2.
//   ACLK = n/a, MCLK = SMCLK = UCLK0 = XT2 = 8MHz
//   Baud rate divider with 8Mhz XTAL @19200 = 8MHz/19200 = 416.66 ~ 417 (01A0h)
//   //* An external 8MHz XTAL on X2IN X2OUT is required for XT2CLK *//
//   //* Min Vcc required varies with MCLK frequency - refer to datasheet *//
//
//
//                        MSP430F149
//                     -----------------
//               /|\|                  XT2IN|-
//                | |                       | 8Mhz
//                --|RST            XT2OUT|-
//                  |                       |
//                  |                   P3.4|------------>
//                  |                       | 19200 - 8N1
//                  |                   P3.5|<------------
//
//
//   M. Buccini
//   Texas Instruments Inc.
//   Feb 2005
//   Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
   volatile unsigned int i;

   BoardConfig(0xb8);
   P3SEL |= 0x30;                                // P3.4,5 = USART0 TXD/RXD
   WDTCTL = WDTPW + WDTHOLD;                     // Stop WDT

   BCSCTL1 &= ~XT2OFF;                           // XT2on

   do
   {
   IFG1 &= ~OFIFG;                               // Clear OSCFault flag
   for (i = 0xFF; i > 0; i--);                   // Time for flag to set
```

```
  }
  while ((IFG1 & OFIFG));                   // OSCFault flag still set?

  BCSCTL2 |= SELM_2 + SELS;                 // MCLK = SMCLK = XT2 (safe)

  ME1 |= UTXE0 + URXE0;                     // Enable USART0 TXD/RXD
  UCTL0 |= CHAR;                            // 8-bit character
  UTCTL0 |= SSEL1;                          // UCLK = SMCLK
  UBR00 = 0xA0;                             // 8Mhz/19200 ~ 417
  UBR10 = 0x01;                             //
  UMCTL0 = 0x00;                            // no modulation
  UCTL0 &= ~SWRST;                          // Initialize USART state machine
  IE1 |= URXIE0;                            // Enable USART0 RX interrupt

  _BIS_SR(LPM0_bits + GIE);                 // Enter LPM0 w/ interrupt
}

#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
  while (!(IFG1 & UTXIFG0));                // USART0 TX buffer ready?
  TXBUF0 = RXBUF0;                          // RXBUF0 to TXBUF0
}
```

```
//*******************************************************************************
//  MSP-FET430P140 Demo - USART0, UART 115200 Echo ISR, XT2 HF XTAL ACLK
//
//  Description: Echo a received character, RX ISR used. Normal mode is LPM0,
//  USART0 RX interrupt triggers TX Echo. Though not required, MCLK= XT2.
//  ACLK = n/a, MCLK = SMCLK = UCLK0 = XT2 = 8MHz
//  Baud rate divider with 8Mhz XTAL = 8000000/115200 = 0069 (0045h)
//  //* An external 8MHz XTAL on X2IN X2OUT is required for XT2CLK *//
//  //* Min Vcc required varies with MCLK frequency - refer to datasheet *//
//
//
//                      MSP430F149
//                   -----------------
//             /|\|                  XT2IN|-
//              | |                       | 8Mhz
//              --|RST             XT2OUT|-
//                |                       |
//                |                   P3.4|------------>
//                |                       | 115200 - 8N1
//                |                   P3.5|<------------
//
//
```

```
//   M. Buccini
//   Texas Instruments Inc.
//   Feb 2005
//   Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"


void main(void)
{
   volatile unsigned int i;

   BoardConfig(0xb8);
   WDTCTL = WDTPW + WDTHOLD;                         // Stop WDT
   P3SEL |= 0x30;                                    // P3.4,5 = USART0 TXD/RXD

   BCSCTL1 &= ~XT2OFF;                               // XT2on

   do
   {
   IFG1 &= ~OFIFG;                                   // Clear OSCFault flag
   for (i = 0xFF; i > 0; i--);                       // Time for flag to set
   }
   while ((IFG1 & OFIFG));                           // OSCFault flag still set?

   BCSCTL2 |= SELM_2 + SELS;                         // MCLK = SMCLK = XT2 (safe)
   ME1 |= UTXE0 + URXE0;                             // Enable USART0 TXD/RXD
   UCTL0 |= CHAR;                                    // 8-bit character
   UTCTL0 |= SSEL1;                                  // UCLK = SMCLK
   UBR00 = 0x45;                                     // 8MHz 115200
   UBR10 = 0x00;                                     // 8MHz 115200
   UMCTL0 = 0x00;                                    // 8MHz 115200 modulation
   UCTL0 &= ~SWRST;                                  // Initialize USART state machine
   IE1 |= URXIE0;                                    // Enable USART0 RX interrupt

   _BIS_SR(LPM0_bits + GIE);                         // Enter LPM0 w/ interrupt
}

#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
   while (!(IFG1 & UTXIFG0));                        // USART0 TX buffer ready?
   TXBUF0 = RXBUF0;                                  // RXBUF0 to TXBUF0
```

```
}
//*****************************************************************************************
//   MSP-FET430P140 Demo - ADC12, Sample A0, Set P3.4 if A0 > 0.5*AVcc
//
//   Description: A single sample is made on A0 with reference to AVcc.
//   Software sets ADC10SC to start sample and conversion - ADC12SC
//   automatically cleared at EOC. ADC12 internal oscillator times sample (16x)
//   and conversion. In Mainloop MSP430 waits in LPM0 to save power until ADC12
//   conversion complete, ADC12_ISR will force exit from LPM0 in Mainloop on
//   reti. If A0 > 0.5*AVcc, P3.4 set, else reset.
//
//                            MSP430F149
//                        -----------------
//                    /|\|                 XIN|-
//                     | |                    |
//                     --|RST                 XOUT|-
//                       |                    |
//           Vin-->|P6.0/A0           P3.4|--> LED
//
//   Dasheng
//   LiTian Electronic Inc.
//   Feb 2008
//   Built with IAR Embedded Workbench Version: 3.42A
//*****************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"


void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;                          // Stop WDT
   BoardConfig(0xb8);
   ADC12CTL0 = SHT0_2 + ADC12ON;                      // Set sampling time, turn on ADC12
   ADC12CTL1 = SHP;                                   // Use sampling timer
   ADC12IE = 0x01;                                    // Enable interrupt
   ADC12CTL0 |= ENC;                                  // Conversion enabled
   P6SEL |= 0x01;                                     // P6.0 ADC option select
   P3DIR |= BIT4;                                     // P3.4 output

   for (;;)
   {
      ADC12CTL0 |= ADC12SC;                           // Sampling open
      _BIS_SR(CPUOFF + GIE);                          // LPM0, ADC12_ISR will force exit
   }
```

```
}

// ADC12 interrupt service routine
#pragma vector=ADC_VECTOR
__interrupt void ADC12_ISR (void)
{
      if (ADC12MEM0 < 0x7FF)
         P3OUT &= ~BIT4;                          // Clear P3.4 LED off
      else
         P3OUT |= BIT4;                          // Set P3.4 LED on
      _BIC_SR_IRQ(CPUOFF);                        // Clear CPUOFF bit from 0(SR)
}
```

```
//*******************************************************************************************************
//   MSP-FET430P140 Demo - ADC12, Using the Internal Reference
//
//   Description: This example shows how to use the internal reference of the ADC12.
//   It uses the internal 2.5V reference and performs a single conversion
//   on channel A0. The conversion results are stored in ADC12MEM0. Test by
//   applying a voltage to channel A0, then setting and running to a break point
//   at the "_NOP()" instruction. To view the conversion results, open an ADC12
//   register window in debugger and view the contents of ADC12MEM0.
//
//
//                         MSP430F149
//                     ---------------
//                    |                 |
//        Vin -->|P6.0/A0            |
//                    |                 |
//
//
//   M. Mitchell
//   Texas Instruments Inc.
//   Feb 2005
//   Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************************************

#include   <msp430x14x.h>
#include   "BoardConfig.h"

void main(void)
{
   volatile unsigned int i;
   WDTCTL = WDTPW+WDTHOLD;                        // Stop watchdog timer
   BoardConfig(0xb8);
   P6SEL |= 0x01;                                 // Enable A/D channel A0
```

```
  ADC12CTL0 = ADC12ON+SHT0_2+REFON+REF2_5V; // Turn on and set up ADC12
  ADC12CTL1 = SHP;                          // Use sampling timer
  ADC12MCTL0 = SREF_1;                      // Vr+=Vref+

  for ( i=0; i<0x3600; i++)                 // Delay for reference start-up
  {
  }

  ADC12CTL0 |= ENC;                         // Enable conversions

  while (1)
  {
    ADC12CTL0 |= ADC12SC;                   // Start conversion
    while ((ADC12IFG & BIT0)==0);
    _NOP();                                 // SET BREAKPOINT HERE
  }
}
```

```
//*******************************************************************************
//  MSP-FET430P140 Demo - ADC12, Sample A10 Temp, Set P1.0 if Temp ++ ~2C
//
//  Description: Use ADC12 and the integrated temperature sensor to detect
//  temperature gradients. The temperature sensor output voltage is sampled
//  ~ every 80ms and compared with the defined delta values using an ISR.
//  (ADC12OSC/256)/ determines sample time which needs to be greater than
//  30us for temperature sensor.
//  ADC12 is operated in repeat-single channel mode with the sample and
//  convert trigger sourced from Timer_A CCR1. The ADC12MEM0_IFG at the end
//  of each converstion will trigger an ISR.
//  ACLK = n/a, MCLK = SMCLK = default DCO ~ 800k, ADC12CLK = ADC12OSC
//
//                        MSP430F149
//                    -----------------
//              /|\|                  XIN|-
//               | |                     |
//               --|RST                XOUT|-
//                 |                     |
//                 |A10                P3.4|-->LED
//
//  A. Dannenberg
//  Texas Instruments Inc.
//  Feb 2005
//  Built with IAR Embedded Workbench Version: 3.21A
//*******************************************************************************

#include  <msp430x14x.h>
```

```
#include    "BoardConfig.h"

#define ADCDeltaOn              12                      // ~2 Deg C delta

static unsigned int FirstADCVal;                    // holds 1st ADC result

void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;                            // Stop watchdog
   BoardConfig(0xb8);
   P3OUT = 0x00;                                        // Clear P1
   P3DIR = BIT4;                                        // P3.4 as output
   ADC12CTL1 = SHS_1 + SHP + CONSEQ_2;                  // TA trig., rpt conv.
   ADC12MCTL0 = SREF_1 + INCH_10;                       // Channel A10, Vref+
   ADC12IE = 0x01;                                      // Enable ADC12IFG.0
   ADC12CTL0 = SHT0_8 + REF2_5V + REFON + ADC12ON + ENC; // Config ADC12
   TACCTL1 = OUTMOD_4;                                  // Toggle on EQU1 (TAR = 0)
   TACTL = TASSEL_2 + MC_2;                             // SMCLK, cont-mode
   while (!(0x01 & ADC12IFG));                          // First conversion?
   FirstADCVal = ADC12MEM0;                             // Read out 1st ADC value
   _BIS_SR(LPM0_bits + GIE);                            // Enter LPM0 w/ interrupt
}

#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR (void)
{
   if (ADC12MEM0 <= FirstADCVal + ADCDeltaOn)
         P3OUT &= ~BIT4;                                // LED off
   else P3OUT |= BIT4;                                  // LED on
}
```

```
/************************************************************
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）
------------------------------------------------------------
测试说明：聆听蜂鸣器发声的音调变化。
************************************************************/
#include <msp430.h>
#include "BoardConfig.h"

uchar step = 0xff;

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

```
    uchar i;

    WDTCTL = WDTPW + WDTHOLD;                   //关狗
    BoardConfig(0xb0);                          //关数码管、流水灯和电平转换

    /*------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
        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

    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
    TBCCTL0 |= CCIE;
    TBCTL |= TBSSEL_1 + ID_3 + MC_1;        //时钟源 ACLK/8，up mode

    P6DIR |= BIT7;                              //蜂鸣器对应 IO 设置为输出
    P2DIR = 0xff;
    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)                          //up mode
      TACTL |= MC_1;
    step++;
    switch(step)
    {
    case    0:  TACCR0 = 5000;  P2OUT = ~1;   break;      // 100Hz
    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;                  //  循环播放
    }
}
```

```
//《祝你平安》对应的编码
const unsigned char SONG[]=
{
    0x26,0x20,0x20,0x20,0x20,0x20,0x26,0x10,0x20,
    0x10,0x20,0x80,0x26,0x20,0x30,0x20,0x30,0x20,
    0x39,0x10,0x30,0x10,0x30,0x80,0x26,0x0,0x20,
    0x20,0x20,0x20,0x1c,0x20,0x20,0x80,0x2b,0x20,
    0x26,0x20,0x20,0x20,0x2b,0x10,0x26,0x10,0x2b,
    0x80,0x26,0x20,0x30,0x20,0x30,0x20,0x39,0x10,
    0x26,0x10,0x26,0x60,0x40,0x10,0x39,0x10,0x26,
    0x20,0x30,0x20,0x30,0x20,0x39,0x10,0x6,0x10,
    0x26,0x80,0x26,0x20,0x2b,0x10,0x2b,0x10,0x2b,
    0x20,0x30,0x10,0x39,0x10,0x26,0x10,0x2b,0x10,
    0x2b,0x20,0x2b,0x40,0x40,0x20,0x20,0x10,0x20,
```

```
    0x10,0x2b,0x10,0x26,0x30,0x30,0x80,0x18,0x20,
    0x18,0x20,0x26,0x20,0x20,0x20,0x20,0x40,0x26,
    0x20,0x2b,0x20,0x30,0x20,0x30,0x20,0x1c,0x20,
    0x20,0x20,0x20,0x80,0x1c,0x20,0x1c,0x20,0x1c,
    0x20,0x30,0x20,0x30,0x60,0x39,0x10,0x30,0x10,
    0x20,0x20,0x2b,0x10,0x26,0x10,0x2b,0x10,0x26,
    0x10,0x26,0x10,0x2b,0x10,0x2b,0x80,0x18,0x20,
    0x18,0x20,0x26,0x20,0x20,0x20,0x20,0x60,0x26,
    0x10,0x2b,0x20,0x30,0x20,0x30,0x20,0x1c,0x20,
    0x20,0x20,0x20,0x80,0x26,0x20,0x30,0x10,0x30,
    0x10,0x30,0x20,0x39,0x20,0x26,0x10,0x2b,0x10,
    0x2b,0x20,0x2b,0x40,0x40,0x10,0x40,0x10,0x20,
    0x10,0x20,0x10,0x2b,0x10,0x26,0x30,0x30,0x80,
    0x00
};

/*********************************************
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   "BoardConfig.h"
#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;

    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

    BoardConfig(0xf8);                              //关闭数码管、流水灯、电平转换

    //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;
                }
            }
        }
    }
}
```

```
/*****************************************************
程序功能：用扫描方式读取四个独立式按键的键值，同时将
          按键的键值在数码管上显示出来
-----------------------------------------------
测试说明：按动 K1~k4 四个按键，观察数码管显示
*****************************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"

#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 )
{
    uchar temp,keyval = 0;

    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0x88);              //打开数码管，关闭流水灯和电平转换

    P1DIR = BIT7;                    //设置 P1.0~P.3 为输入状态，P.7 为输出
    P1OUT = 0;
    P3DIR |= BIT4;                   //设置 P3.4 为输出状态
    P3OUT |= BIT4;                   //P3.4 输出 1
    P4DIR = 0xff;
    P5DIR = 0xff;
    P4OUT = 0x3f;
    P5OUT = 0xf7;
    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;
                while(keyin != 0x0f);     //等待按键被放开
                switch(temp)       //转换键值
                {
                case 0x0e:
```

```
                        keyval = 1;break;
                case 0x0d:
                        keyval = 2;break;
                case 0x0b:
                        keyval = 3;break;
                case 0x07:
                        keyval = 4;break;
                default:
                        keyval = 0;break;
                }
                P4OUT = scandata[keyval]; //用一位数码管显示
                P3OUT &= ~BIT4;                 //P3.4 连接的 LED 闪烁一下
                delay();delay();
                P3OUT |= BIT4;
            }
        }
    }
}
/*****************************************
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--);
}
```

```
/*************************************************
程序功能：用中断方式读取四个独立式按键的键值，同时将
          按键的键值在数码管上显示出来
-------------------------------------------------
测试说明：按动 K1~k4 四个按键，观察数码管显示
*************************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"

#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};
```

```
uchar KeyVal = 0;     //  按键的键值

void delay(void);

/********************主函数********************/
void main( void )
{
    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0x88);              //打开数码管，关闭流水灯和电平转换

    P1IES = 0x0f;                   // P1.0~P1.3 选择下降沿中断
    P1IE = 0x0f;                    //  打开中断使能
    P1DIR = BIT7;                    //设置 P1.0~P.3 为输入状态，P.7 为输出
    P1OUT = 0;
    P4DIR = 0xff;
    P5DIR = 0xff;
    P4OUT = 0x3f;
    P5OUT = 0xf7;
    _EINT();                         //打开全局中断控制位
    while(1)
    {
        LPM1;
        P4OUT = scandata[KeyVal];
    }
}
/*******************************************
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--);
}
/*******************************************
函数名称：PORT1_ISR
功      能：P1 端口的中断服务函数
参      数：无
返回值  ：无
*******************************************/
#pragma vector=PORT1_VECTOR
__interrupt void    PORT1_ISR(void)
```

```
{
    if(P1IFG & 0x0f)
    {
        switch(P1IFG)
        {
        case 0x01:
                    if(keyin == 0x0e)     //如果是第一个按键被按下
                    {
                        delay();
                        if(keyin == 0x0e)
                        {
                            while(keyin != 0x0f);            //等待按键放开
                            KeyVal = 1;
                            LPM1_EXIT;
                            P1IFG = 0;
                            return;
                        }
                    }
        case 0x02:
                    if(keyin == 0x0d)     //如果是第二个按键被按下
                    {
                        delay();
                        if(keyin == 0x0d)
                        {
                            while(keyin != 0x0f);            //等待按键放开
                            KeyVal = 2;
                            LPM1_EXIT;
                            P1IFG = 0;
                            return;
                        }
                    }
        case 0x04:
                    if(keyin == 0x0b)     //如果是第三个按键被按下
                    {
                        delay();
                        if(keyin == 0x0b)
                        {
                            while(keyin != 0x0f);            //等待按键放开
                            KeyVal = 3;
                            LPM1_EXIT;
                            P1IFG = 0;
                            return;
                        }
                    }
        case 0x08:
```

```
                    if(keyin == 0x07)     //如果是第四个按键被按下
                    {
                        delay();
                        if(keyin == 0x07)
                        {
                            while(keyin != 0x0f);           //等待按键放开
                            KeyVal = 4;
                            LPM1_EXIT;
                            P1IFG = 0;
                            return;
                        }
                    }
        default:
                    while(keyin != 0x0f);           //等待按键放开
                    //KeyVal = 0;
                    //LPM1_EXIT;
                    P1IFG = 0;
                    return;
        }
    }
}
```

```
/*************************************************************
程序功能：用中断方式读取四个独立式按键的键值，同时用
          D1~D4 四个 LED 的点亮来指示 k1~K4 四个按键的按下
-------------------------------------------------------
测试说明：按动 K1~k4 四个按键，观察 LED 是否正确点亮
*************************************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"

#define keyin       (P1IN & 0x0f)

uchar KeyVal = 0;     //  按键的键值

/*********************主函数*********************/
void main( void )
{
    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0xb0);              //打开流水灯，关闭数码管和电平转换

    P1IES = 0x0f;                   // P1.0~P1.3 选择下降沿中断
    P1IE    = 0x0f;                 //  打开中断使能
    P1DIR = BIT7;                    //设置 P1.0~P.3 为输入状态，P.7 为输出
    P1OUT = 0;
    P2DIR = 0xff;
```

```
    P2OUT = 0xff;
    _EINT();                            //打开全局中断控制位
    while(1)
    {
        LPM1;
        P2OUT = ~(1 << (KeyVal - 1));
    }
}
/*****************************************
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--);
}
/*****************************************
函数名称：PORT1_ISR
功    能：P1 端口的中断服务函数
参    数：无
返回值  ：无
*****************************************/
#pragma vector=PORT1_VECTOR
__interrupt void  PORT1_ISR(void)
{
    if(P1IFG & 0x0f)
    {
        switch(P1IFG)
        {
        case 0x01:
                    if(keyin == 0x0e)    //如果是第一个按键被按下
                    {
                        delay();
                        if(keyin == 0x0e)
                        {
                            while(keyin != 0x0f);          //等待按键放开
                            KeyVal = 1;
                            LPM1_EXIT;
                            P1IFG = 0;
                            return;
                        }
```

```
            }
case 0x02:
            if(keyin == 0x0d)     //如果是第二个按键被按下
            {
                delay();
                if(keyin == 0x0d)
                {
                    while(keyin != 0x0f);           //等待按键放开
                    KeyVal = 2;
                    LPM1_EXIT;
                    P1IFG = 0;
                    return;
                }
            }
case 0x04:
            if(keyin == 0x0b)     //如果是第三个按键被按下
            {
                delay();
                if(keyin == 0x0b)
                {
                    while(keyin != 0x0f);           //等待按键放开
                    KeyVal = 3;
                    LPM1_EXIT;
                    P1IFG = 0;
                    return;
                }
            }
case 0x08:
            if(keyin == 0x07)     //如果是第四个按键被按下
            {
                delay();
                if(keyin == 0x07)
                {
                    while(keyin != 0x0f);           //等待按键放开
                    KeyVal = 4;
                    LPM1_EXIT;
                    P1IFG = 0;
                    return;
                }
            }
default:
            while(keyin != 0x0f);           //等待按键放开
            P1IFG = 0;
            return;
}
```

```
    }
}
```

```
/************************************************************
程序功能：用按键控制蜂鸣器发音的音调。
          K1 按下后用 2KHz 方波驱动蜂鸣器
          K2 按下后用 4KHz 方波驱动蜂鸣器
          K3 按下后用 6KHz 方波驱动蜂鸣器
          K4 按下后停止发音
------------------------------------------------------------
测试说明：按动 K1~k4 四个按键，聆听蜂鸣器发声频率
************************************************************/
#include <msp430.h>
#include "BoardConfig.h"


#define keyin       (P1IN & 0x0f)

uchar step = 0xff;

void main( void )
{
    uchar i;

    WDTCTL = WDTPW + WDTHOLD;                   //关狗
    BoardConfig(0xb0);                          //关数码管和电平转换,打开流水灯

    /*------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

    TACTL |= TASSEL_2 + ID_3 + MC_1;        //计数时钟选择 SMLK=8MHz，1/8 分频后为 1MHz

    P1IES = 0x0f;                               // P1.0~P1.3 选择下降沿中断
    P1IE = 0x0f;                                //  打开中断使能
    P1DIR = BIT7;                               //设置 P1.0~P.3 为输入状态，P.7 为输出
    P1OUT = 0;
    P2DIR = 0xff;
    P2OUT = 0xff;
    P6DIR |= BIT7;                              //蜂鸣器对应 IO 设置为输出
```

```
    P6OUT |= BIT7;

    _EINT();
    LPM1;
}
// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P6OUT ^= BIT7;                            // Toggle P6.7
}
/*******************************************
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;
    uchar i;

    for(i = 7; i > 0; i--)
    {
        for(tmp = 12000;tmp > 0;tmp--);
    }
}
/*******************************************
函数名称：PORT1_ISR
功      能：P1 端口的中断服务函数
参      数：无
返回值  ：无
*******************************************/
#pragma vector=PORT1_VECTOR
__interrupt void   PORT1_ISR(void)
{
    if(P1IFG & 0x0f)
    {
        switch(P1IFG)
        {
        case 0x01:
                    if(keyin == 0x0e)     //如果是第一个按键被按下
                    {
                        delay();
                        if(keyin == 0x0e)
```

```
            {
                while(keyin != 0x0f);           //等待按键放开
                TACCR0 = 250;                       // 2KHz
                TACCTL0 |= CCIE;
                P2OUT = 0xfe;
                P1IFG = 0;
                return;
            }
        }
case 0x02:
        if(keyin == 0x0d)     //如果是第二个按键被按下
        {
            delay();
            if(keyin == 0x0d)
            {
                while(keyin != 0x0f);           //等待按键放开
                TACCR0 = 125;                       // 4KHz
                TACCTL0 |= CCIE;
                P2OUT = 0x0fd;
                P1IFG = 0;
                return;
            }
        }
case 0x04:
        if(keyin == 0x0b)     //如果是第三个按键被按下
        {
            delay();
            if(keyin == 0x0b)
            {
                while(keyin != 0x0f);           //等待按键放开
                TACCR0 = 83;                        // 6KHz
                TACCTL0 |= CCIE;
                P2OUT = 0xfb;
                P1IFG = 0;
                return;
            }
        }
case 0x08:
        if(keyin == 0x07)     //如果是第四个按键被按下
        {
            delay();
            if(keyin == 0x07)
            {
                while(keyin != 0x0f);           //等待按键放开
                P6OUT |= BIT7;
```

```
                                TACCTL0 &= ~CCIE;                   //停止发声
                                P2OUT = 0xf7;
                                P1IFG = 0;
                                return;
                            }
                        }
            default:
                        while(keyin != 0x0f);           //等待按键放开
                        P1IFG = 0;
                        return;
            }
        }

        P1IFG = 0;
}
```

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

void Init_Keypad(void);
void Check_Key(void);
void delay();
void Key_Event(void);

#include   <msp430x14x.h>
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)
{
    P1DIR = 0xf0;          //P1.0~P1.3 设置为输入状态, P1.4~P1.7 设置为输出状态
    P1OUT |= 0xf0;         // P1.4~P1.7 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
    {
        P1OUT = 0xf0;                           //P1.4~P1.7 输出全 1
        P1OUT -= tmp1;                          //P1.4~p1.7 输出四位中有一个为 0
        tmp1 >>=1;
        if ((P1IN & 0x0f) < 0x0f)               //是否 P1IN 的 P1.0~P1.3 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((P1IN & 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;

    P1OUT &= 0x00;                    // 设置 P1OUT 全为 0，等待按键输入
    tmp = P1IN;                       // 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();
    }
}

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

```
---------------------------------------------
跳线设置：将跳线座 J3 上的短路帽拔下
---------------------------------------------
测试说明：按动 K1~K16 按键，观察数码管显示
**********************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"
#include "Keypad.h"
#include "gdata.h"

/********************主函数********************/
void main(void)
{
    BoardConfig(0x88);                  //打开数码管，关闭流水灯和电平转换
    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

    P4DIR = 0xff;                       //设置 P4，P5 的 IO 方向为输出
    P5DIR = 0xff;
    P4OUT = 0x00;                       //设置 P4，P5 的输出初值
    P5OUT = 0xff;
    _EINT();                            //打开全局中断
    Init_Keypad();
    while(1)
    {
        Key_Event();

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

```
    P5OUT = 0xff;
    P4OUT = scandata[Dispbuf[cnt]];              //输出段选信号
    P5OUT &= ~(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;
}
```

```
void DispStr(unsigned char x,unsigned char y,unsigned char *ptr);
void DispNChar(unsigned char x,unsigned char y, unsigned char n,unsigned char *ptr);
void LocateXY(unsigned char x,unsigned char y);
void Disp1Char(unsigned char x,unsigned char y,unsigned char data);
void LcdReset(void);
void LcdWriteCommand(unsigned char cmd,unsigned char chk);
void LcdWriteData( unsigned char data );
void WaitForEnable(void);
void Delay5ms(void);

#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         P3DIR
#define CLR_RS P3OUT&=~BIT0;       //RS = P3.0
#define SET_RS P3OUT|=BIT0;
#define CLR_RW P3OUT&=~BIT1;    //RW = P3.1
#define SET_RW P3OUT|=BIT1;
#define CLR_EN P3OUT&=~BIT2;     //EN = P3.2
#define SET_EN P3OUT|=BIT2;
/*************************************************
函数名称：DispStr
功      能：让液晶从某个位置起连续显示一个字符串
参      数：x--位置的列坐标
            y--位置的行坐标
            ptr--指向字符串存放位置的指针
返回值  ：无
*************************************************/
void DispStr(uchar x,uchar y,uchar *ptr)
{
    uchar *temp;
```

```
    uchar i,n = 0;

    temp = ptr;
    while(*ptr++ != '\0')     n++;      //计算字符串有效字符的个数

    for (i=0;i<n;i++)
    {
        Disp1Char(x++,y,temp[i]);
        if (x == 0x0f)
        {
            x    = 0;
            y ^= 1;
        }
    }
}
/*******************************************
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 |= 0x07;                    //控制线端口设为输出状态
    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
    _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
    _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--;
    }
}

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

```
程序功能：动态显示文字“welcome!”
-----------------------------------------------
测试说明：观察液晶显示
*************************************************/
#include   <msp430.h>
#include   "BoardConfig.h"
#include   "Cry1602.h"

uchar *s1 = "welcome!";

void main( void )
{
    uchar i;
    WDTCTL = WDT_ADLY_250;                    //间隔定时器，定时 16ms
    BoardConfig(0xb8);                        //关闭数码管、流水灯和电平转换

    LcdReset();
    DispStr(4,0,s1);
    LocateXY(0,9);                            //确定写入字符的显示位置
    LcdWriteCommand(0x07, 1);                 //整体显示左移
    for(i = 12; i > 0; i--)
    {
        LcdWriteData(0x20);
        //延时 250ms
        IFG1 &= ~WDTIFG;
        while(!(IFG1 & WDTIFG));
        IFG1 &= ~WDTIFG;
    }
    while(1)
    {
        LcdWriteCommand(0x05, 1);             //整体显示右移

        for(i = 24; i > 0; i--)
        {
            LcdWriteData(0x20);
            //延时 250ms
            IFG1 &= ~WDTIFG;
            while(!(IFG1 & WDTIFG));
            IFG1 &= ~WDTIFG;
        }


        LcdWriteCommand(0x07, 1);             //整体显示左移

        for(i = 24; i > 0; i--)
```

```
        {
            LcdWriteData(0x20);
            //延时 250ms
            IFG1 &= ~WDTIFG;
            while(!(IFG1 & WDTIFG));
            IFG1 &= ~WDTIFG;
        }
    }
}
```

```
/*****************************************************
程序功能：静态显示各种字符
-----------------------------------------------------
测试说明：观察液晶显示
*****************************************************/
#include   <msp430.h>
#include   "BoardConfig.h"
#include   "Cry1602.h"

uchar shuzi[] = {"0123456789"};
uchar zimu1[] = {"abcdefghigklmnopqrstuvwxyz"};
uchar zimu2[] = {"ABCDEFGHIGKLMNOPQRSTUVWXYZ"};
uchar *fuhao = "~`!@#$%^&*()_+-=/|,.、;'<>?:""";
uchar *jieshu = "This is the end!";

/************************主函数*************************/
void main( void )
{
    uchar i;
    WDTCTL = WDT_ADLY_1000;                    //间隔定时器，定时 1000ms
    BoardConfig(0xb8);                         //关闭数码管、流水灯和电平转换

    LcdReset();
    DispNChar(3,0,10,shuzi);
    //延时 2s
    for(i = 0; i < 3; i++)
    {

        IFG1 &= ~WDTIFG;
        while(!(IFG1 & WDTIFG));
        IFG1 &= ~WDTIFG;
    }
    LcdWriteCommand(0x01, 1);                  //清除显示
    DispNChar(0,0,26,zimu1);
    //延时 2s
    for(i = 0; i < 3; i++)
```

```
    {

        IFG1 &= ~WDTIFG;
        while(!(IFG1 & WDTIFG));
        IFG1 &= ~WDTIFG;
    }
    LcdWriteCommand(0x01, 1);                //清除显示
    DispNChar(0,0,26,zimu2);
    //延时 2s
    for(i = 0; i < 3; i++)
    {

        IFG1 &= ~WDTIFG;
        while(!(IFG1 & WDTIFG));
        IFG1 &= ~WDTIFG;
    }
    LcdWriteCommand(0x01, 1);                //清除显示
    DispStr(0,0,fuhao);
    //延时 2s
    for(i = 0; i < 3; i++)
    {

        IFG1 &= ~WDTIFG;
        while(!(IFG1 & WDTIFG));
        IFG1 &= ~WDTIFG;
    }
    LcdWriteCommand(0x01, 1);                //清除显示
    DispStr(0,0,jieshu);
    while(1);
}
```

```
/*****************************************************
程序功能：在 1602 液晶上显示用 MCU 的 TimerA 模拟的数字
          式实时时钟。
-----------------------------------------------------
测试说明：观察液晶显示
*****************************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"
#include "cry1602.h"
#include "clock.h"

uchar tishi[]={"Current Time is"};     //提示信息
extern uchar second,minute,hour,hour0;
extern uchar pmin,phour,ps1;
```

```
#define SetTime(H,M,S) {second=S;minute=M;hour=H;hour0=H;}

/*************************主函数************************/
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0xb8);              //关闭 74LVC4245，数码管和流水灯

    CCR0 = 32768 - 1;               //设置定时器 A 的中断时间为 1S
    TACTL = TASSEL_1 + MC_1;       //计数时钟 ACLK, 增计数模式
    CCTL0 |= CCIE;                  //使能 CCR0 比较中断
    _EINT();                        //打开全局中断

    //请在此处设置正确时间
    SetTime(8,26,50);               //顺序：时，分，秒，格式：BCD 码

    LcdReset();
    DispNChar(0,0,15,tishi);        //显示提示文字
    Disp1Char(6,1,':');             //显示字符 :
    Disp1Char(9,1,':');

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

/*******************************************
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)
{
    Clock();
    LPM3_EXIT;
}
```

```
void Delay_1ms(void);
void Delay(unsigned int n);
void Write_Cmd(unsigned char cod);
void Write_Data(unsigned char dat);
void Ini_Lcd(void);
```

```
#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
}

/*****************************************************************
程序功能：在 12864 液晶上显示 ASCII 常用字符
-----------------------------------------------------------
跳线设置：将跳线座 J5 的 1 脚(+)和 2 脚短接，选择并行数据传输方式
-----------------------------------------------------------
测试说明：观察液晶显示
*****************************************************************/
#include   "msp430.h"
```

```
#include   "BoardConfig.h"
#include   "cry12864.h"

/*****************************主函数*****************************/
void main( void )
{
    uint i,j;
    uchar tishi[] = {"This is the end!"};

    WDTCTL = WDTPW + WDTHOLD;       //关狗
    BoardConfig(0xb8);

    Ini_Lcd();                      //初始化液晶

    Write_Cmd(0x80);                //写第一行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(0x30 + i);         //显示 0x30~0x3f 对应的字符
    Write_Cmd(0x90);                //写第一行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(0x40 + i);         //显示 0x40~0x4f 对应的字符
    Write_Cmd(0x88);                //写第一行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(0x50 + i);         //显示 0x50~0x5f 对应的字符
    Write_Cmd(0x98);                //写第一行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(0x60 + i);         //显示 0x60~0x6f 对应的字符

    for(i = 1000; i > 0; i--)       //延时一会
    {
        for(j = 1000; j > 0; j--)
          _NOP();
    }

    Write_Cmd(0x01);                //清屏

    Write_Cmd(0x80);                //写第一行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(0x70 + i);         //显示 0x30~0x3f 对应的字符

    Write_Cmd(0x88);                //写第三行的显示地址
    for(i = 0; i < 16; i++)
      Write_Data(tishi[i]);         //显示 0x30~0x3f 对应的字符

    LPM4;
}
```

```
void DelayNus(unsigned int n);
unsigned char Init_18B20(void);
void Write_18B20(unsigned char wdata);
unsigned char Read_18B20(void);
void Skip(void);
void Convert(void);
void Read_SP(void);
unsigned int ReadTemp(void);
unsigned int Do1Convert(void);


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

#define DQ1 P1OUT |= BIT6
#define DQ0 P1OUT &= ~BIT6

/*****************************************
函数名称：DelayNus
功      能：实现 N 个微秒的延时
参      数：n--延时长度
返回值   ：无
说明     ：定时器 A 的计数时钟是 1MHz，CPU 主频 8MHz
           所以通过定时器延时能够得到极为精确的
           us 级延时
*****************************************/
void DelayNus(uint n)
{
    CCR0 = n;
    TACTL |= MC_1;                  //增计数到 CCR0
    while(!(TACTL & BIT0));     //等待
    TACTL &= ~MC_1;                 //停止计数
    TACTL &= ~BIT0;                 //清除中断标志
}
/*****************************************
函数名称：Init_18B20
功      能：对 DS18B20 进行复位操作
参      数：无
返回值   ：初始化状态标志：1--失败，0--成功
*****************************************/
uchar Init_18B20(void)
{
    uchar Error;
```

```
    _DINT();
    DQ0;
    DelayNus(500);
    DQ1;
    DelayNus(55);
    P1DIR &=~ BIT6;
    _NOP();
    if(P1IN & BIT6)
    {
        Error = 1;              //初始化失败
        P1DIR |= BIT6;
    }
    else
    {
        Error = 0;              //初始化成功
        P1DIR |= BIT6;
        DQ1;
    }
    _EINT();

    DelayNus(400);

    return Error;
}
/*******************************************
函数名称：Write_18B20
功      能：向 DS18B20 写入一个字节的数据
参      数：wdata--写入的数据
返回值  ：无
*******************************************/
void Write_18B20(uchar wdata)
{
    uchar i;

    _DINT();
    for(i = 0; i < 8;i++)
    {
        DQ0;
        DelayNus(6);                //延时 6us
        if(wdata & 0X01)      DQ1;
        else                   DQ0;
        wdata >>= 1;
        DelayNus(50);               //延时 50us
        DQ1;
        DelayNus(10);               //延时 10us
```

```
    }
    _EINT();
}
/*********************************************
函数名称：Read_18B20
功    能：从 DS18B20 读取一个字节的数据
参    数：无
返回值  ：读出的一个字节数据
*********************************************/
uchar Read_18B20(void)
{
    uchar i;
    uchar temp = 0;

    _DINT();
    for(i = 0;i < 8;i++)
    {
        temp >>= 1;
        DQ0;
        DelayNus(6);                //延时 6us
        DQ1;
        DelayNus(8);                //延时 9us
        P1DIR &= ~BIT6;
        _NOP();
        if(P1IN & BIT6)     temp |= 0x80;
        DelayNus(45);               //延时 45us
        P1DIR |= BIT6;
        DQ1;
        DelayNus(10);               //延时 10us
    }
    _EINT();

    return  temp;
}

/*********************************************
函数名称：Skip
功    能：发送跳过读取产品 ID 号命令
参    数：无
返回值  ：无
*********************************************/
void Skip(void)
{
    Write_18B20(0xcc);
}
```

```
/*******************************************
函数名称：Convert
功　　能：发送温度转换命令
参　　数：无
返回值　：无
*******************************************/
void Convert(void)
{
    Write_18B20(0x44);
}
/*******************************************
函数名称：Read_SP
功　　能：发送读 ScratchPad 命令
参　　数：无
返回值　：无
*******************************************/
void Read_SP(void)
{
    Write_18B20(0xbe);
}
/*******************************************
函数名称：ReadTemp
功　　能：从 DS18B20 的 ScratchPad 读取温度转换结果
参　　数：无
返回值　：读取的温度数值
*******************************************/
uint ReadTemp(void)
{
    uchar temp_low;
    uint   temp;

    temp_low = Read_18B20();          //读低位
    temp = Read_18B20();        //读高位
    temp = (temp<<8) | temp_low;

    return   temp;
}
/*******************************************
函数名称：ReadTemp
功　　能：控制 DS18B20 完成一次温度转换
参　　数：无
返回值　：测量的温度数值
*******************************************/
uint Do1Convert(void)
{
```

```
    uchar i;

    do
    {
        i = Init_18B20();
    }
    while(i);
    Skip();
    Convert();
    for(i = 20;i > 0;i--)
        DelayNus(60000); //延时 800ms 以上
    do
    {
        i = Init_18B20();
    }
    while(i);
    Skip();
    Read_SP();
    return ReadTemp();
}

/************************************************
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 "BoardConfig.h"
#include "DS18B20.h"

//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};
//数码管位选变量
uchar cnt = 0;

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

    BoardConfig(0x88);                          //打开数码管

    //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;
    P5DIR = 0xff;
    P4OUT = 0x00;
    P5OUT = 0xff;

    //设置 DS18B20 的 IO 状态
    P1DIR |= BIT6;
    P1OUT |= BIT6;
    //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)
{
```

```
    P5OUT = 0xff;
    P4OUT = scandata[dN[5-cnt]];
    if(cnt==1) P4OUT |= 0x80;     //在第二位显示小数点
    P5OUT &= ~(1<<cnt);

    cnt++;
    if(cnt == 6) cnt = 0;
}
/*******************************************
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;
        }
    }
}
```

```
void DelayNus(unsigned int n);
unsigned char Init_18B20(void);
void Write_18B20(unsigned char wdata);
unsigned char Read_18B20(void);
void Skip(void);
void Convert(void);
void Read_SP(void);
unsigned int ReadTemp(void);
unsigned int Do1Convert(void);
```

```
void DispStr(unsigned char x,unsigned char y,unsigned char *ptr);
void DispNChar(unsigned char x,unsigned char y, unsigned char n,unsigned char *ptr);
void LocateXY(unsigned char x,unsigned char y);
void Disp1Char(unsigned char x,unsigned char y,unsigned char data);
void LcdReset(void);
void LcdWriteCommand(unsigned char cmd,unsigned char chk);
void LcdWriteData( unsigned char data );
void WaitForEnable(void);
void Delay5ms(void);
```

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

#define DQ1 P1OUT |= BIT6
#define DQ0 P1OUT &= ~BIT6

/*******************************************
函数名称：DelayNus
功      能：实现 N 个微秒的延时
参      数：n--延时长度
```

```
返回值  ：无
说明    ：定时器 A 的计数时钟是 1MHz，CPU 主频 8MHz
          所以通过定时器延时能够得到极为精确的
          us 级延时
*******************************************/
void DelayNus(uint n)
{
    CCR0 = n;
    TACTL |= MC_1;                  //增计数到 CCR0
    while(!(TACTL & BIT0));     //等待
    TACTL &= ~MC_1;                 //停止计数
    TACTL &= ~BIT0;                 //清除中断标志
}
/*******************************************
函数名称：Init_18B20
功      能：对 DS18B20 进行复位操作
参      数：无
返回值  ：初始化状态标志：1--失败，0--成功
*******************************************/
uchar Init_18B20(void)
{
    uchar Error;

    _DINT();
    DQ0;
    DelayNus(500);
    DQ1;
    DelayNus(55);
    P1DIR &=~ BIT6;
    _NOP();
    if(P1IN & BIT6)
    {
        Error = 1;              //初始化失败
        P1DIR |= BIT6;
    }
    else
    {
        Error = 0;              //初始化成功
        P1DIR |= BIT6;
        DQ1;
    }
    _EINT();

    DelayNus(400);
```

```
    return Error;
}
/*********************************************
函数名称：Write_18B20
功    能：向 DS18B20 写入一个字节的数据
参    数：wdata--写入的数据
返回值  ：无
*********************************************/
void Write_18B20(uchar wdata)
{
    uchar i;

    _DINT();
    for(i = 0; i < 8;i++)
    {
        DQ0;
        DelayNus(6);                //延时 6us
        if(wdata & 0X01)      DQ1;
        else                   DQ0;
        wdata >>= 1;
        DelayNus(50);               //延时 50us
        DQ1;
        DelayNus(10);               //延时 10us
    }
    _EINT();
}
/*********************************************
函数名称：Read_18B20
功    能：从 DS18B20 读取一个字节的数据
参    数：无
返回值  ：读出的一个字节数据
*********************************************/
uchar Read_18B20(void)
{
    uchar i;
    uchar temp = 0;

    _DINT();
    for(i = 0;i < 8;i++)
    {
        temp >>= 1;
        DQ0;
        DelayNus(6);                //延时 6us
        DQ1;
        DelayNus(8);                //延时 9us
```

```
            P1DIR &= ~BIT6;
            _NOP();
            if(P1IN & BIT6)     temp |= 0x80;
            DelayNus(45);                   //延时 45us
            P1DIR |= BIT6;
            DQ1;
            DelayNus(10);                   //延时 10us
        }
        _EINT();

        return    temp;
}

/*******************************************
函数名称：Skip
功      能：发送跳过读取产品 ID 号命令
参      数：无
返回值   ：无
*******************************************/
void Skip(void)
{
        Write_18B20(0xcc);
}
/*******************************************
函数名称：Convert
功      能：发送温度转换命令
参      数：无
返回值   ：无
*******************************************/
void Convert(void)
{
        Write_18B20(0x44);
}
/*******************************************
函数名称：Read_SP
功      能：发送读 ScratchPad 命令
参      数：无
返回值   ：无
*******************************************/
void Read_SP(void)
{
        Write_18B20(0xbe);
}
/*******************************************
函数名称：ReadTemp
```

```
功      能：从 DS18B20 的 ScratchPad 读取温度转换结果
参      数：无
返回值  ：读取的温度数值
*******************************************/
uint ReadTemp(void)
{
    uchar temp_low;
    uint  temp;

    temp_low = Read_18B20();        //读低位
    temp = Read_18B20();        //读高位
    temp = (temp<<8) | temp_low;

    return  temp;
}
/*******************************************
函数名称：ReadTemp
功      能：控制 DS18B20 完成一次温度转换
参      数：无
返回值  ：测量的温度数值
*******************************************/
uint Do1Convert(void)
{
    uchar i;

    do
    {
        i = Init_18B20();
    }
    while(i);
    Skip();
    Convert();
    for(i = 20;i > 0;i--)
        DelayNus(60000); //延时 800ms 以上
    do
    {
        i = Init_18B20();
    }
    while(i);
    Skip();
    Read_SP();
    return ReadTemp();
}
```

```
#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          P3DIR
#define CLR_RS P3OUT&=~BIT0;       //RS = P3.0
#define SET_RS P3OUT|=BIT0;
#define CLR_RW P3OUT&=~BIT1;    //RW = P3.1
#define SET_RW P3OUT|=BIT1;
#define CLR_EN P3OUT&=~BIT2;     //EN = P3.2
#define SET_EN P3OUT|=BIT2;
/*********************************************
函数名称：DispStr
功      能：让液晶从某个位置起连续显示一个字符串
参      数：x--位置的列坐标
            y--位置的行坐标
            ptr--指向字符串存放位置的指针
返回值  ：无
*********************************************/
void DispStr(uchar x,uchar y,uchar *ptr)
{
    uchar *temp;
    uchar i,n = 0;

    temp = ptr;
    while(*ptr++ != '\0')     n++;       //计算字符串有效字符的个数

    for (i=0;i<n;i++)
    {
        Disp1Char(x++,y,temp[i]);
        if (x == 0x0f)
        {
            x    = 0;
            y ^= 1;
        }
    }
}
/*********************************************
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 |= 0x07;                    //控制线端口设为输出状态
    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
    _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
    _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--;
    }
}
```

```
/*************************************************************
程序功能：读取 DS18B20 进行温度测量以后的结果并在 1602 液晶上显示
-------------------------------------------------------------
测试说明：观察显示温度数值。
*************************************************************/

#include   "msp430.h"
#include   "BoardConfig.h"
#include   "cry1602.h"
#include "DS18B20.h"

//要显示的 6 位温度数字
uchar dN[6];

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

    WDTCTL = WDTPW + WDTHOLD;                 //关狗
    BoardConfig(0xb8);
```

```
    /*------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 DS18B20 的 IO 状态
    P1DIR |= BIT6;
    P1OUT |= BIT6;
    //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();

    LcdReset();
    DispStr(0,0,"Temperature is:");
    //循环读数显示
    while(1)
    {
        Disp_Numb(Do1Convert());
        Disp1Char(4,1,dN[5]+0x30);
        Disp1Char(5,1,dN[4]+0x30);
        Disp1Char(6,1,0x2e);              //0x2e 是小数点对应的 ASCII 码值
        Disp1Char(7,1,dN[3]+0x30);
        Disp1Char(8,1,dN[2]+0x30);
        Disp1Char(9,1,dN[1]+0x30);
        Disp1Char(10,1,dN[0]+0x30);
    }
}

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

```
unsigned char shuzi[]={"0123456789"};
unsigned char tishi[]={"The time is : "};

//秒，分，时，日，月，星期，年
unsigned char wdata[7]={0x30,0x46,0x20,0x14,0x08,0x02,0x07};
```

```
unsigned char rdata[7];
//秒，分，时，日，月，星期，年，控制
unsigned char bwdata[8]={0x30,0x34,0x20,0x24,0x11,0x02,0x10,0x00};
unsigned char brdata[8];
//读写 RAM 中的数据
unsigned char rwdata[31]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
                          18,19,20,21,22,23,24,25,26,27,28,29,30,31};
unsigned char rrdata[31];
```

---

ds1302

```
void delay(unsigned int time);
void Reset_DS1302(void);
void Write1Byte(unsigned char wdata);
unsigned char Read1Byte(void);
void W_Data(unsigned char addr, unsigned char wdata);
unsigned char R_Data(unsigned char addr);
void BurstWrite1302(unsigned char *ptr);
void BurstRead1302(unsigned char *ptr);
void BurstWriteRAM(unsigned char *ptr);
void BurstReadRAM(unsigned char *ptr);
void Set_DS1302(unsigned char *ptr);
void Get_DS1302(unsigned char *ptr);
```

1602

```
void DispNchar(unsigned char x,unsigned char y, unsigned char n,unsigned char *ptr);
void LocateXY(unsigned char x,unsigned char y);
void Disp1Char(unsigned char x,unsigned char y,unsigned char data);
void LcdReset(void);
void LcdWriteCommand(unsigned char cmd,unsigned char chk);
void LcdWriteData( unsigned char data );
void WaitForEnable(void);
void Delay5ms(void);
void Delay400ms(void);
```

1302.c

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

/***************宏定义****************/
#define DS_RST    BIT7              //DS_RST = P2.7
#define DS_SCL    BIT5              //DS_SCL = P2.5
#define DS_SDA    BIT6               //DS_SDA = P2.6
#define DS_RST_IN P2DIR &= ~DS_RST
#define DS_RST_OUT P2DIR |= DS_RST
#define DS_RST0 P2OUT &= ~DS_RST
#define DS_RST1 P2OUT |= DS_RST
#define DS_SCL_IN P2DIR &= ~DS_SCL
```

```
#define DS_SCL_OUT P2DIR |= DS_SCL
#define DS_SCL0 P2OUT &= ~DS_SCL
#define DS_SCL1 P2OUT |= DS_SCL
#define DS_SDA_IN P2DIR &= ~DS_SDA
#define DS_SDA_OUT P2DIR |= DS_SDA
#define DS_SDA0 P2OUT &= ~DS_SDA
#define DS_SDA1 P2OUT |= DS_SDA
#define DS_SDA_BIT P2IN & 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;
    }
}
```

1602.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          P3DIR
#define CLR_RS P3OUT &= ~BIT0;       //RS = P3.0
#define SET_RS P3OUT |= BIT0;
#define CLR_RW P3OUT &= ~BIT1;       //RW = P3.1
#define SET_RW P3OUT |= BIT1;
#define CLR_EN P3OUT &= ~BIT2;       //EN = P3.2
#define SET_EN P3OUT |= BIT2;
/*****************************************
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 |= 0x07;                    //控制线端口设为输出状态
    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
    _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
    _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--);
    }
}
```

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

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

    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0xf8);              //关闭数码管、流水灯、电平转换

    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);
}
```

```
}
```

Key.c

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

#define keyin       (P1IN & 0x0f)

/*******************************************
函数名称：delay
功      能：用于消抖的延时
参      数：无
返回值  ：无
*******************************************/
void delay_10ms(void)
{
    uint tmp;

    for(tmp = 0x3fff;tmp > 0;tmp--);
}
/*******************************************
函数名称：Key4Scan
功      能：扫描四个独立式按键
参      数：无
返回值  ：键值
*******************************************/
uchar Key4Scan(void)
{
    uchar temp,keyval;

    if(keyin != 0x0f)           //如果有键被按下
    {
        delay_10ms();                   //延时消抖
        keyval = 0;
        if(keyin != 0x0f)     //再次检测按键状态
        {
            temp=keyin;
            while(keyin != 0x0f);     //等待按键被放开
            switch(temp)       //转换键值
            {
            case 0x0e:
                        keyval = 1;break;
            case 0x0d:
                        keyval = 2;break;
            case 0x0b:
```

```
                        keyval = 3;break;
                case 0x07:
                        keyval = 4;break;
                default:
                        keyval = 0;break;
                }
        }
    }
    else
        keyval = 0;

    return keyval;

}
```

1602.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          P3DIR
#define CLR_RS P3OUT&=~BIT0;       //RS = P3.0
#define SET_RS P3OUT|=BIT0;
#define CLR_RW P3OUT&=~BIT1;    //RW = P3.1
#define SET_RW P3OUT|=BIT1;
#define CLR_EN P3OUT&=~BIT2;     //EN = P3.2
#define SET_EN P3OUT|=BIT2;
/*******************************************
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 |= 0x07;                    //控制线端口设为输出状态
    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
    _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
    _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--);
    }
}
```

1302.c

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

/**************宏定义***************/
#define DS_RST    BIT7              //DS_RST = P2.7
#define DS_SCL    BIT5              //DS_SCL = P2.5
#define DS_SDA    BIT6               //DS_SDA = P2.6
#define DS_RST_IN P2DIR &= ~DS_RST
#define DS_RST_OUT P2DIR |= DS_RST
#define DS_RST0 P2OUT &= ~DS_RST
#define DS_RST1 P2OUT |= DS_RST
#define DS_SCL_IN P2DIR &= ~DS_SCL
```

```
#define DS_SCL_OUT P2DIR |= DS_SCL
#define DS_SCL0 P2OUT &= ~DS_SCL
#define DS_SCL1 P2OUT |= DS_SCL
#define DS_SDA_IN P2DIR &= ~DS_SDA
#define DS_SDA_OUT P2DIR |= DS_SDA
#define DS_SDA0 P2OUT &= ~DS_SDA
#define DS_SDA1 P2OUT |= DS_SDA
#define DS_SDA_BIT P2IN & 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;
    }
}
```

```
/*****************************************************
程序功能：在 1602 液晶上显示一个数字日历，可以通过
          四个按键来设置各种参数
-----------------------------------------------------
测试说明：（1）按 K1 键进入设置模式并可以选择更改参
              数的位置，
          （2）按 K2 键单方向增加数值
          （3）按 K3 键放弃当前修改回到工作模式
          （4）按 K4 键保存当前数值回到工作模式
          实际按键观察测试。
*****************************************************/
#include <msp430x14x.h>
#include "BoardConfig.h"
#include "cry1602.h"
#include "DS1302.h"

//顺序：秒，分，时，日，月，星期，年；格式：BCD
uchar times[7];
//液晶显示数字编码
uchar shuzi[] = {"0123456789"};
//游标位置变量
uchar PP = 0;
//是否处于修改状态标志，1--是，0--否
uchar cflag = 0;

uchar Key4Scan(void);
void ShowTime(void);
/****************主函数****************/
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    BoardConfig(0xb8);              // 关闭数码管、流水灯、电平转换

    P1DIR = 0x80;                    //P1.7 设置为输出，其余为输入
    P1OUT = 0x00;

    Reset_DS1302();                 //初始化 DS1302
    LcdReset();                     //初始化液晶
    while(1)
    {
```

```
if(!cflag)
{
    Get_DS1302(times);              //获取时间数据
    ShowTime();                      //转换显示
}

switch(Key4Scan())
{
case 0x01:
            switch(PP++)        //确定游标地址
            {
            case 0: LocateXY(4,0);break;
            case 1: LocateXY(7,0);break;
            case 2: LocateXY(10,0);break;
            case 3: LocateXY(13,0);break;
            case 4: LocateXY(4,1);break;
            case 5: LocateXY(7,1);break;
            case 6: LocateXY(10,1);break;
            default:break;
            }
            LcdWriteCommand(0x0f, 1);    //打开游标
            if(PP == 7) PP = 0;
            cflag = 1;              //标志置位
            break;
case 0x02:
            if(cflag)
            {
                switch(PP)
                {
                case 1:             //年
                        times[6]++;
                        if((times[6]&0x0f) == 0x0a)
                        {
                            times[6] += 0x06;
                        }
                        if(times[6] > 0x99)
                        {
                            times[6] = 0x00;
                        }
                        Disp1Char(3,0,shuzi[times[6]>>4]);
                        Disp1Char(4,0,shuzi[times[6]&0x0f]);
                        LocateXY(4,0);
                        break;
                case 2:             //月
                        times[4]++;
```

```
            if((times[4]&0x0f) == 0x0a)
            {
                times[4] += 0x06;
            }
            if(times[4] > 0x12)
            {
                times[4] = 0x01;
            }
            Disp1Char(6,0,shuzi[times[4]>>4]);
            Disp1Char(7,0,shuzi[times[4]&0x0f]);
            LocateXY(7,0);
            break;
case 3:             //日
            times[3]++;
            if((times[3]&0x0f) == 0x0a)
            {
                times[3] += 0x06;
            }
            if(times[3] > 0x31)
            {
                times[3] = 0x01;
            }
            Disp1Char(9,0,shuzi[times[3]>>4]);
            Disp1Char(10,0,shuzi[times[3]&0x0f]);
            LocateXY(10,0);
            break;
case 4:             //周
            times[5]++;
            if((times[5]&0x0f) == 0x08)
            {
                times[5] = 0x01;
            }
            Disp1Char(13,0,shuzi[times[5]]);
            LocateXY(13,0);
            break;
case 5:             //时
            times[2]++;
            if((times[2]&0x0f) == 0x0a)
            {
                times[2] += 0x06;
            }
            if(times[2] > 0x23)
            {
                times[2] = 0x00;
            }
```

```
                    Disp1Char(3,1,shuzi[times[2]>>4]);
                    Disp1Char(4,1,shuzi[times[2]&0x0f]);
                    LocateXY(4,1);
                    break;
                case 6:             //分
                    times[1]++;
                    if((times[1]&0x0f) == 0x0a)
                    {
                        times[1] += 0x06;
                    }
                    if(times[1] > 0x59)
                    {
                        times[1] = 0x00;
                    }
                    Disp1Char(6,1,shuzi[times[1]>>4]);
                    Disp1Char(7,1,shuzi[times[1]&0x0f]);
                    LocateXY(7,1);
                    break;
                case 0:             //时
                    times[0]++;
                    if((times[0]&0x0f) == 0x0a)
                    {
                        times[0] += 0x06;
                    }
                    if(times[0] > 0x59)
                    {
                        times[0] = 0x00;
                    }
                    Disp1Char(9,1,shuzi[times[0]>>4]);
                    Disp1Char(10,1,shuzi[times[0]&0x0f]);
                    LocateXY(10,1);
                    break;
                default:
                    break;

                }
            }
            break;
case 0x03:
            if(cflag)
            {
                cflag = 0;
                PP = 0;
                LcdWriteCommand(0x0c, 1);     //关闭游标
            }
```

```
                    break;
            case 0x04:
                    if(cflag)
                    {
                        cflag = 0;
                        PP = 0;
                        LcdWriteCommand(0x0c, 1);     //关闭游标
                        Set_DS1302(times);
                    }
                    break;
            default:
                    break;
            }
        }
}
/*****************************************
函数名称：ShowTime
功      能：将 DS1302 的时间转换成 10 进制显示
参      数：无
返回值  ：无
*****************************************/
void ShowTime(void)
{
    uchar h1[14];           //第 1 行显示数据
    uchar h2[8];            //第 2 行显示数据

    h1[0] = shuzi[2];
    h1[1] = shuzi[0];
    h1[2] = shuzi[times[6]>>4];       //年
    h1[3] = shuzi[times[6]&0x0f];
    h1[4] = 0x2d;           //"-"
    h1[5] = shuzi[times[4]>>4];       //月
    h1[6] = shuzi[times[4]&0x0f];
    h1[7] = 0x2d;           //"-"
    h1[8] = shuzi[times[3]>>4];       //日
    h1[9] = shuzi[times[3]&0x0f];
    h1[10] = 0x20;          //" "
    h1[11] = 0x2a;          //"*"
    h1[12] = shuzi[times[5]];         //周
    h1[13] = 0x2a;          //"*"
    DispNChar(1,0,14,h1);               //在第一行显示

    h2[0] = shuzi[times[2]>>4];       //时
    h2[1] = shuzi[times[2]&0x0f];
    h2[2] = 0x3a;           //":"
```

```
    h2[3] = shuzi[times[1]>>4];         //分
    h2[4] = shuzi[times[1]&0x0f];
    h2[5] = 0x3a;               //":"
    h2[6] = shuzi[times[0]>>4];         //秒
    h2[7] = shuzi[times[0]&0x0f];
    DispNChar(3,1,8,h2);                  //在第二行显示
}
```