MSP430F6638_LED跑马灯

张立为

版本管理

- 修改记录

版本号.	作者	描述	修改日期
V01	张立为		2012-11-2

- 审核记录

版本号.	职务	签名	修改日期

DY 德研電科

轮廓

- GPIO的基本概念
- LED跑马灯硬件实现
- LED跑马灯简单软件编程实现

GPIO基本概念

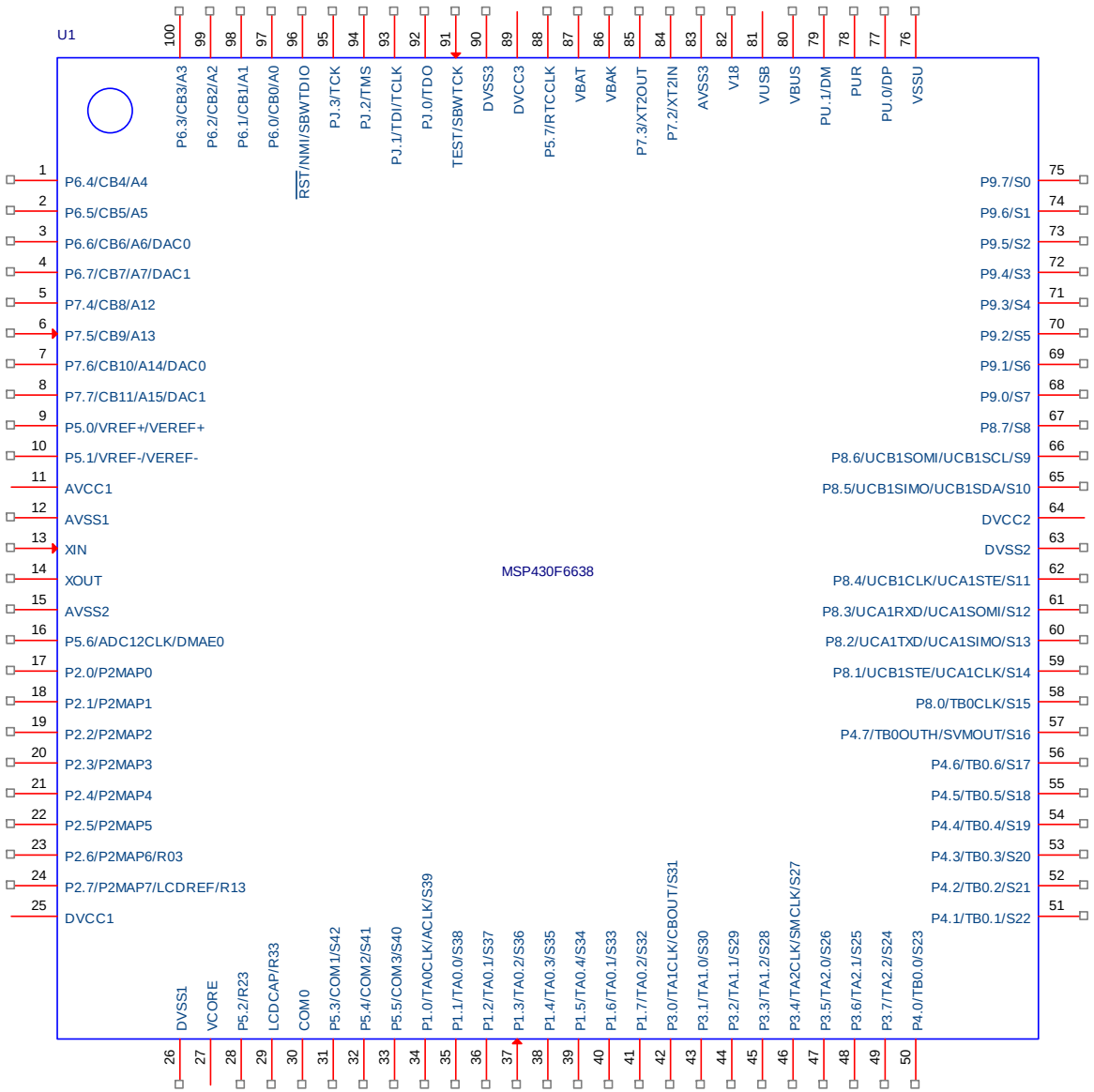
- GPIO——通用输入输出

GPIO是MCU数据输入输出的基本模块，可以实现MCU与外部电路进行数据交换。

- GPIO功能

数字输入/输出，并行/串行通讯，存储器扩展

MSP430F6638通用输入输出端口

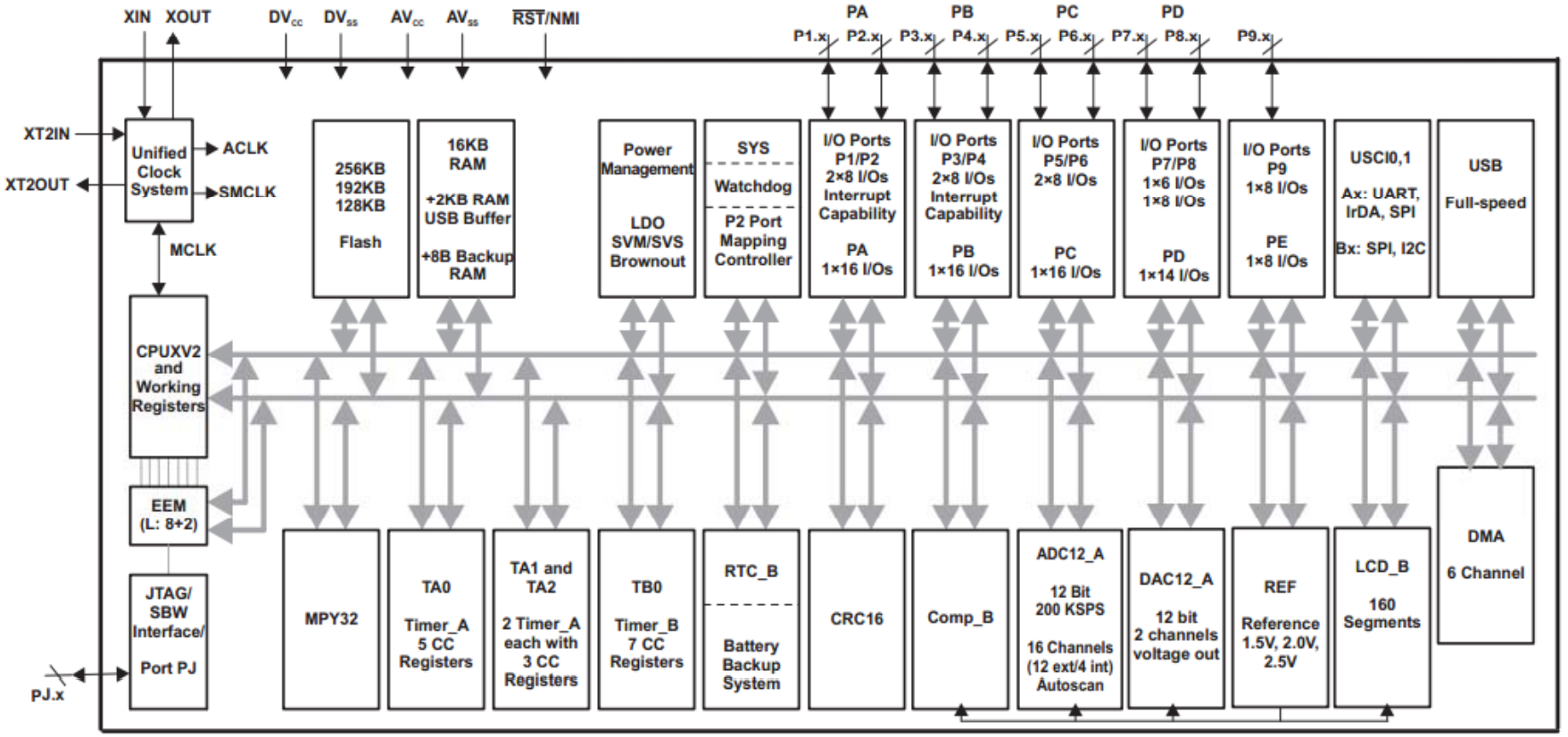


MSP430F663x介绍

- MSP430F663x是微控制器系列产品，配置有一个高性能12位模数（A/D）转换器，比较器，2个通用串行通信接口（USCI），USB2.0，硬件乘法器，DMA，4个16位计时器，具有报警功能的实时时钟模块，LCD驱动器和多达74I/O引脚。
- 这款芯片的典型应用包括模拟和数字传感器系统，数字电机控制，遥控，恒温器，数字时钟，手持仪表等。

设备	闪存 (KB)	SRAM (KB) [1]	Timer_A [2]	Timer_B [3]	USCI		ADC12_A (Ch)	DAC12_A (Ch)	Comp_B (Ch)	I/O	封装类型
					通道A: UART/IrDA/ SPI	通道 B: SPI/I^2C					
MSP430F6638	256	16 + 2	5, 3, 3	7	2	2	12 ext / 4 int	2	12	74	100 PZ, 113 ZQW

功能块图解


XIN
XOUT
DVcc
DVss
AVcc
AVss
RST/NMI
PA
PB
PC
PD
P1.x
P2.x
P3.x
P4.x
P5.x
P6.x
P7.x
P8.x
P9.x
XT2IN
XT2OUT
Unified Clock System
ACLK
SMCLK
MCLK
256KB 192KB 128KB Flash
16KB RAM
+2KB RAM USB Buffer
+8B Backup RAM
Power Management
LDO SVM/SVS Brownout
SYS
Watchdog
P2 Port Mapping Controller
I/O Ports P1/P2 2×8 I/Os Interrupt Capability
PA 1×16 I/Os
I/O Ports P3/P4 2×8 I/Os Interrupt Capability
PB 1×16 I/Os
I/O Ports P5/P6 2×8 I/Os
PC 1×16 I/Os
I/O Ports P7/P8 1×6 I/Os 1×8 I/Os
PD 1×14 I/Os
I/O Ports P9 1×8 I/Os
PE 1×8 I/Os
USCI0,1
Ax: UART, IrDA, SPI
Bx: SPI, I2C
USB
Full-speed
CPUXV2 and Working Registers
EEM (L: 8+2)
JTAG/ SBW Interface/
Port PJ
PJ.x
MPY32
TA0
Timer_A 5 CC Registers
TA1 and TA2
2 Timer_A each with 3 CC Registers
TB0
Timer_B 7 CC Registers
RTC_B
Battery Backup System
CRC16
Comp_B
ADC12_A
12 Bit 200 KSPS
16 Channels (12 ext/4 int) Autoscan
DAC12_A
12 bit 2 channels voltage out
REF
Reference 1.5V, 2.0V, 2.5V
LCD_B
160 Segments
DMA
6 Channel

功能块图解

- 由上图可见（此图从MSP430F663x datasheet中截取）GPIO口可简单分为两类（带中断IO和不带中断IO）：

1. PA和PB端口具有中断功能
2. PC、PD、PE端口不具有中断功能

常用寄存器介绍

- 方向选择寄存器 PxDIR　　0：输入；1：输出
- 输入寄存器 PxIN　　检测IO的逻辑状态
- 输出寄存器 PxOUT

  0:低电平输出；1：输出高电平
- 功能选择寄存器 PxSEL

  0：普通I/O口；1：选择第二功能端口
- 上拉/下拉电阻使能寄存器 PxREN

  0：上/下拉功能禁止；1：上/下拉功能使能
- 驱动力选择 PxDS

  0：低驱动力模式；1:高驱动力模式

常用寄存器介绍

- 中断允许寄存器 PxIE

  0：中断允许；1：中断禁止

- 中断边沿选择寄存器 PxIES

  0：上升沿；1：下降沿

- 中断标志寄存器 PxIFG

  0：触发事件没有发生；1：触发事件发生了

GPIO Output Frequency

GPIO输出频率

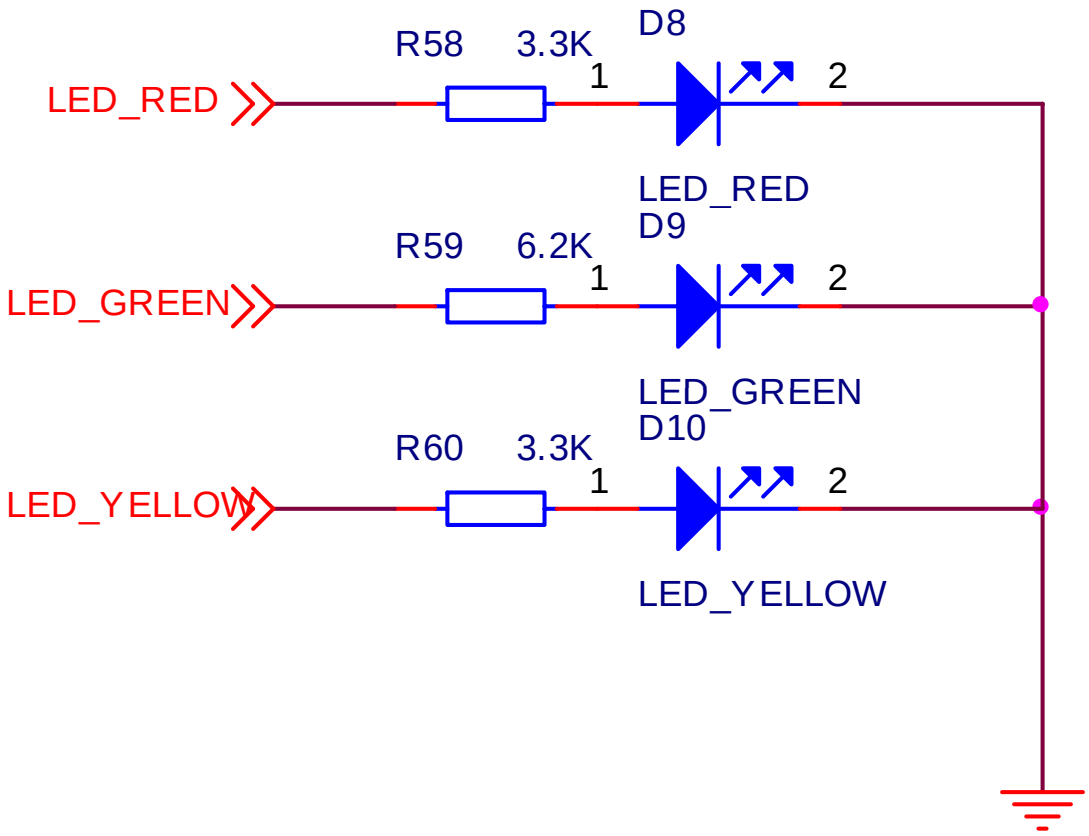
over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	MAX	UNIT
$f_{Px.y}$	Port output frequency (with load)	P3.4/TA2CLK/SMCLK/S27 C_L = 20 pF, R_L = 1 kΩ (1) or 3.2 kΩ(2) (3)	V_{CC} = 1.8 V PMMCOREVx = 0		8	MHz
			V_{CC} = 3 V PMMCOREVx = 3		20	
f_{Port_CLK}	Clock output frequency	P1.0/TA0CLK/ACLK/S39 P3.4/TA2CLK/SMCLK/S27 P2.0/P2MAP0 (P2MAP0 = PM_MCLK) C_L = 20 pF(3)	V_{CC} = 1.8 V PMMCOREVx = 0		8	MHz
			V_{CC} = 3 V PMMCOREVx = 3		20	

(1) Full drive strength of port: A resistive divider with 2 × 0.5 kΩ between V_{CC} and V_{SS} is used as load. The output is connected to the center tap of the divider.
(2) Reduced drive strength of port: A resistive divider with 2 × 1.6 kΩ between V_{CC} and V_{SS} is used as load. The output is connected to the center tap of the divider.
(3) The output voltage reaches at least 10% and 90% V_{CC} at the specified toggle frequency.

LED跑马灯硬件实现

- R、G、Y红绿黄三色发光二极管驱动电路以及接口如下图所示：由MSP430F6638 的P4.1、P4.2、P4.3三个IO驱动控制



限流电阻选材算法

- 上图中三个电阻(R58、R59、R60)的阻值选择：假设VDD为I/O口高电平电压，对于MSP430F6638平台是3.3V，通常，2mA的正向电流 IF足以让LED发光，此时下列根据公式，LED的正向压降VF约为1.2V（具体参数参照购买的LED发光管datasheet）所以，取LED限流电阻阻值为：

$$R = \frac{V_{DD} - V_F}{I_F}$$

- **注：**根据电阻常用规格 和 实际需要 来选择匹配电阻，

LED跑马灯软件简单编程

- LED跑马灯实现方法有很多，这里例举3种不同的方法来说明；

1. 配置寄存器法
2. 直接调用头文件#include <msp430f6638.h>法
3. 通过使用固件库driverlib配置GPIO引脚控制法

配置寄存器法跑马灯

```
 #define P3P4_BASE_Address 0x0220
#define BIT1                (0x0002)
#define BIT2                (0x0004)
#define BIT3                (0x0008)
#define P4DIR  (*((volatile unsigned char*)(P3P4_BASE_Address + 0x05)))
#define P4OUT        (*((volatile unsigned char*)(P3P4_BASE_Address + 0x03)))
void main(void)
{
  volatile unsigned int i;
  volatile unsigned int count=0;

  P4DIR |= BIT1 + BIT2 + BIT3;          // P4.1,P4.2,P4.3 set as output
  while(1)                              // continuous loop
  {
    P4OUT ^= BIT1 + BIT2 + BIT3;        // XOR P4.1,P4.2,P4.3
    for(i=20000;i>0;i--);               // Delay
  }
}
```

调头文件法实现跑马灯

```
#include <msp430f6638.h>

void main(void)
{
  volatile unsigned int i;
  WDTCTL = WDTPW+WDTHOLD;       // Stop WDT
  P4DIR |= BIT1 + BIT2 + BIT3;  // P4.1,P4.2,P4.3 set as output
  while(1)                      // continuous loop
  {
    P4OUT ^= BIT1 + BIT2 + BIT3;  // XOR P4.1,P4.2,P4.3
    for(i=20000;i>0;i--);         // Delay
  }
}
```

使用固件库DRIVERLIB法控制

- 介绍：使用driverlib库的好处：

1、可独立编程各个I/O

2、输入输出任意组合

3、单独配置P1和P2中断和包括某些设备额外的端口中断

4、独立的输入输出数据寄存器

5、可单独配置上拉/下拉电阻

DRIVERLIB中GPIO API函数

- The GPIO pins are configured with

1. GPIO_setAsOutputPin()
2. GPIO_setAsInputPin()
3. GPIO_setAsInputPinWithPullDownresistor()
4. GPIO_setAsInputPinWithPullUpresistor()
5. GPIO_setDriveStrength()
6. GPIO_setAsPeripheralModuleFunctionOutputPin()
7. GPIO_setAsPeripheralModuleFunctionInputPin()

DRIVERLIB中GPIO API函数

- The GPIO interrupts are handled with

1. GPIO_enableInterrupt()
2. GPIO_disbleInterrupt()
3. GPIO_clearInterruptFlag()
4. GPIO_getInterruptStatus()
5. GPIO_interruptEdgeSelect()

DRIVERLIB中GPIO API函数

- The GPIO pin state is accessed with

1. GPIO_setOutputHighOnPin()
2. GPIO_setOutputLowOnPin()
3. GPIO_toggleOutputOnPin()
4. GPIO_getInputPinValue()

- 关于以上函数具体用法请参照Texas Instruments官方www.ti.com手册：
MSP430_DriverLib_Users_Guide-1_20_01_00.pdf

使用固件库DRIVERLIB跑马灯

```
#include "inc/hw_memmap.h"
#include "driverlib/5xx_6xx/gpio.h"
#include "driverlib/5xx_6xx/wdt.h"

void main (void)
{
    volatile unsigned int i;
    WDT_hold(__MSP430_BASEADDRESS_WDT_A__);    //Stop WDT

    //P4.x output
    GPIO_setAsOutputPin(__MSP430_BASEADDRESS_PORT4_R__,
    GPIO_PORT_P4,
        GPIO_PIN1 + GPIO_PIN2 + GPIO_PIN3
        );
```

使用固件库DRIVERLIB跑马灯

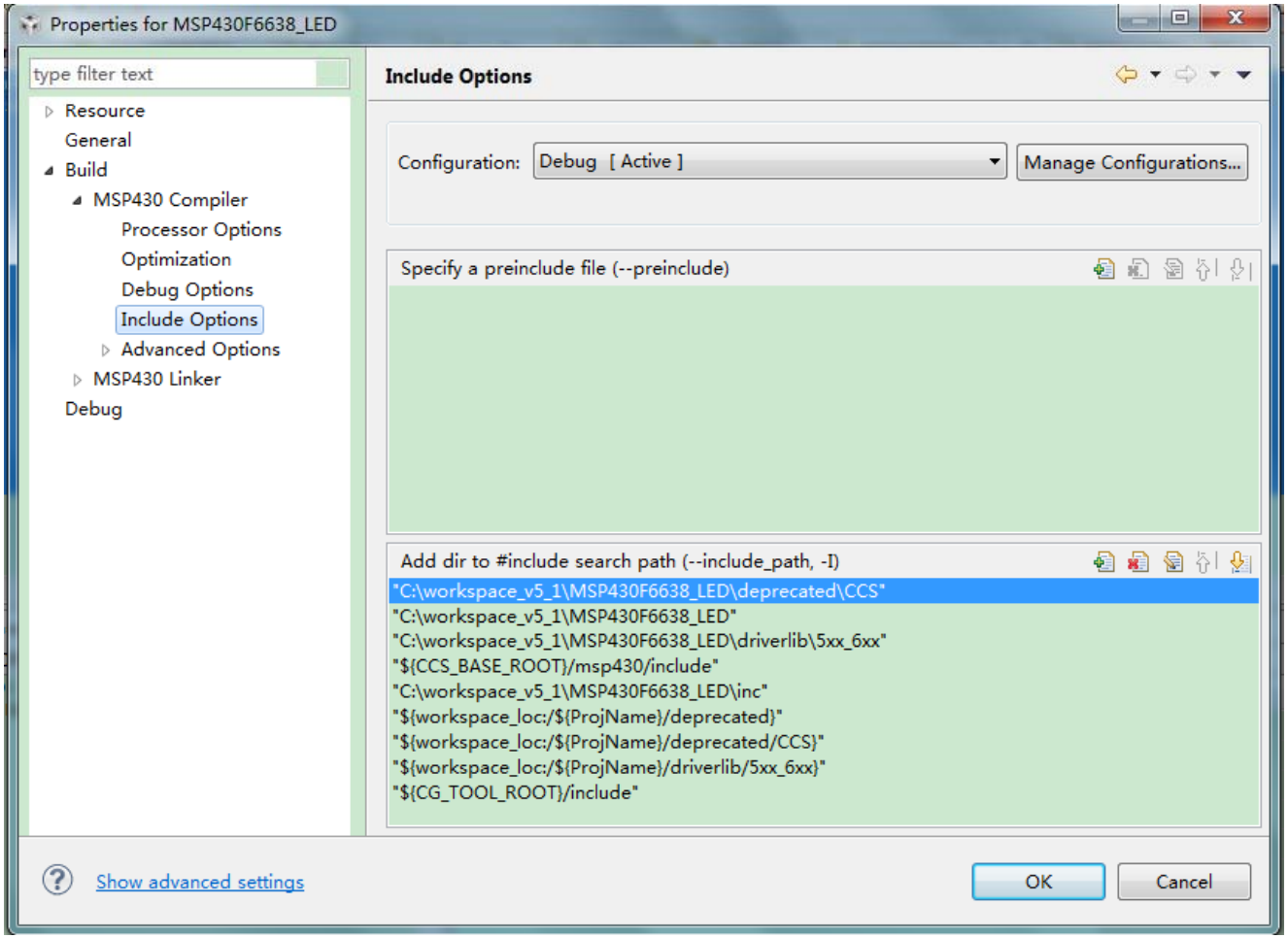
```
while(1)
{
    for(i=50000;i>0;i--);                  // Delay
   //Set all P4pins Low
    GPIO_setOutputLowOnPin(
    __MSP430_BASEADDRESS_PORT4_R__,
    GPIO_PORT_P4,
    GPIO_PIN1 + GPIO_PIN2 + GPIO_PIN3
    );

    for(i=50000;i>0;i--);                  // Delay

    //Set all P4pins HI
    GPIO_setOutputHighOnPin(
    __MSP430_BASEADDRESS_PORT4_R__,
    GPIO_PORT_P4,
    GPIO_PIN1 + GPIO_PIN2 + GPIO_PIN3
    );
    }
}
```

使用固件库DRIVERLIB跑马灯

使用driverlib注意事项：
把driverlib库加进工程之后，注意在Project——Properties——Include Options 选项中添加driverlib所在的路径



采用按键查询控制LED

编程步骤：

1、关闭看门狗

2、初始化按键，把P2.6（USER_2）设置成为输入模式

3、初始化三个LED（P4.1、P4.2、P4.3三个IO），并让三个灯默认为熄灭状态

4、轮询查询USER_2按键是否按下，如果按下，则点亮三个LED灯，否则三个灯熄灭

采用按键查询控制LED

```
#include"msp430f6638.h"
void main(void)
{

WDTCTL = WDTPW+WDTHOLD;                  // Stop WDT

//setting direction
P2DIR &= ~(1<<6);             //setting IO for input  P2.6   (SW4 USER_2)
P4DIR |= (1<<1)|(1<<2)|(1<<3);    //setting IO for output
P4OUT = 0x00;     //led off
while (1)
{
if ((P2IN & 0x40) == 0)          // If key is pressed  USER_2按下（低电平有效）
{
P4OUT = 0x0e;       //led on
}
else
P4OUT = 0X00;      //led off
}
}
```

采用消抖动按键控制LED

添加少许延时，防止按键按下时产生抖动导致的误操作

```
#include"msp430f6638.h"
void main(void)
{
WDTCTL = WDTPW+WDTHOLD;                  // Stop WDT
//setting direction
P2DIR &= ~(1<<6);//setting IO for input  P2.6(SW4 USER_2)
P4DIR |= (1<<1)|(1<<2)|(1<<3);//setting IO for output
P4OUT = 0x00;
while (1)
{
if ((P2IN & 0x40) == 0)//If key is pressed
{
__delay_cycles(5000);    // disappears shakes
if ((P2IN & 0x40) == 0)//If key is pressed
{
P4OUT = 0x0e;//led on
}
}
else
P4OUT = 0X00;//led off
}
}
```

采用按键中断控制LED

编程步骤如下：

1、关闭看门狗
2、初始化三个LED灯，并默认熄灭LED状态
3、使能USER_1、USER_2按键中断
4、清除中断标志
5、使能中断
6、当有按键按下时，产生中断，使三个LED灯状态翻转
7、清除中断标志，等待下次按键中断

采用按键中断控制LED

```
#include"msp430f6638.h"
void interrupt_key(void);
void main(void) {
WDTCTL = WDTPW+WDTHOLD;                 // Stop WDT
interrupt_key();
}

void interrupt_key(void){
P4DIR |= BIT1+BIT2+BIT3;                // P4.1,P4.2,P4.3 set as output
P4OUT &=~(BIT1+BIT2+BIT3);              // set led off
P2IE |= BIT6+BIT7;                      // enable P2.6 ,P2.7 interrupt
P2IFG &= ~(BIT6+BIT7);                  // clean interrupt flag
__enable_interrupt();                   // enable interrupt
while(1);
}
```

采用按键中断控制LED

```
// PORT2 interrupt service routine
#pragma vector=PORT2_VECTOR
__interrupt void port_2(void){
if(P2IN & 0x40){
P4OUT ^= BIT1+BIT2+BIT3;                // XOR P4.1,P4.2,P4.3
}else if(P2IN & 0X80){
P4OUT ^= BIT1+BIT2+BIT3;                // XOR P4.1,P4.2,P4.3
}
P2IFG &=~(BIT6+BIT7);                    // clean interrupt flag
}
```