1.实验题目

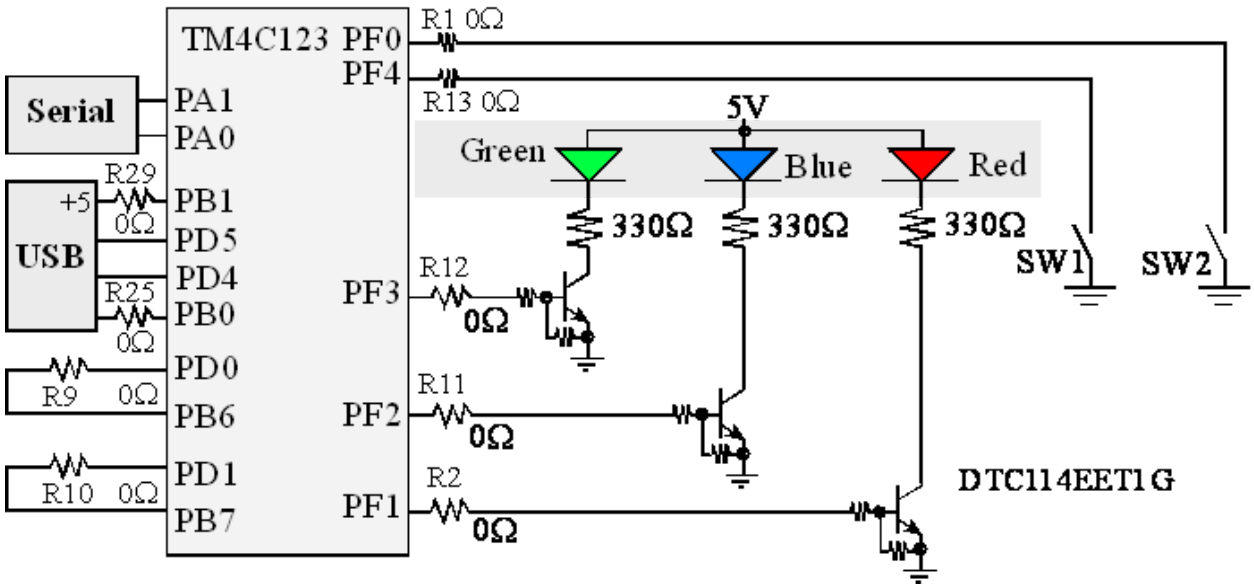
- **工程文件的进阶实验**

2. 实验内容

- **Lab2_1:**LaunchPad Switches and LEDs.
  - 运行源代码解释实验现象
  - 更换输入输出,重做实验
- **Lab2_2:**定义位地址
  - 运行源代码解释实验现象
  - 更换输入输出,重组实验
- **Lab2_3:** Debug mode-select “Use simulator”
  - 运行仿真,分析结果

3. 实验原理

- **Lab2_1**



LaunchPad上LED灯和按键的GPIO口如上图所示.

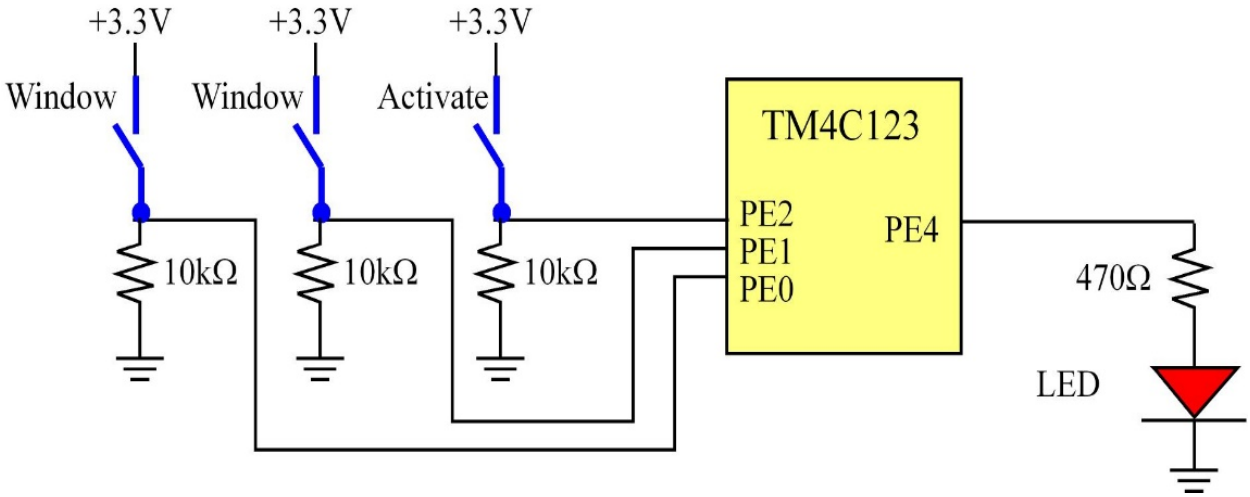
- **Lab2_2**

Port	Base address
PortA	0x40004000
PortB	0x40005000
PortC	0x40006000
PortD	0x40007000
PortE	0x40024000
PortF	0x40025000

If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

实验中需要的GPIO口地址表如上图,使用中根据需要的端口地址进行组合.

- **Lab2_3**



GPIO对应情况如上图(实际仿真中此图为**负逻辑**)。

4. 实验过程

lab2_1:LaunchPad Switches and LEDs.

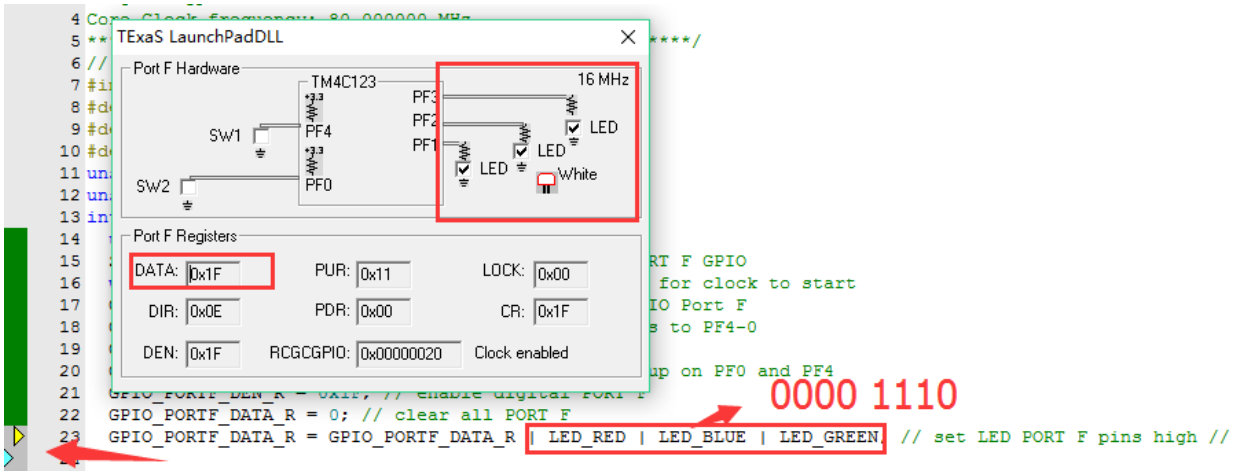
- **代码理解**

```
#include "tm4c123gh6pm.h"
#define LED_RED 0x02
#define LED_BLUE 0x04
#define LED_GREEN 0x08
unsigned long In; // input from PF4
unsigned long Out; // output to PF2 (blue LED)
```

首先这部分是定义部分,其中PF4为左边按键,PF0为右边按键,PF3-1分别为绿蓝红灯.

```
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;// enable PORT F GPIO
while((SYSCTL_PRGPIO_R&0x20)==0){}; // allow time for clock to start
GPIO_PORTF_LOCK_R = 0x4C4F434B;     // 2) unlock GPIO Port F
GPIO_PORTF_CR_R = 0x1F;             // allow changes to PF4-0
GPIO_PORTF_DIR_R = 0x0E;            // PF4,PF0 in, PF3-1 out
GPIO_PORTF_PUR_R = 0x11;            // enable pull-up on PF0 and PF4
GPIO_PORTF_DEN_R = 0x1F;            // enable digital PORT F
GPIO_PORTF_DATA_R = 0;              // clear all PORT F
GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R | LED_RED | LED_BLUE | LED_GREEN;
                                    // set LED PORT F pins high
```

main中while前面部分为Port F的注册先是激活时钟。此处只需要使用到PF0-4，但是其中PF0需要解锁，其他并不需要。最后一行GPIO之前先把所有占用Port F的数据清零，然后使用之前宏定义的三个LED的值也就是的或也就是(0000 1110)在或一次Port F 的数据，这样是初始化PF3-1全亮(此时通过仿真证明)

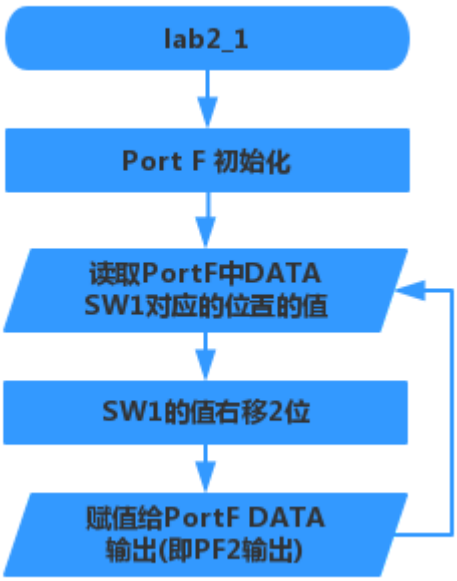


此时我们可以看到DATA为0x1F，因为此时两个sw都是断开状态也就是PF4和PF0为1，此时DATA为0x11(0001 0001)，此时在或上三个LED的0000 1110 得到的就是0x1F。

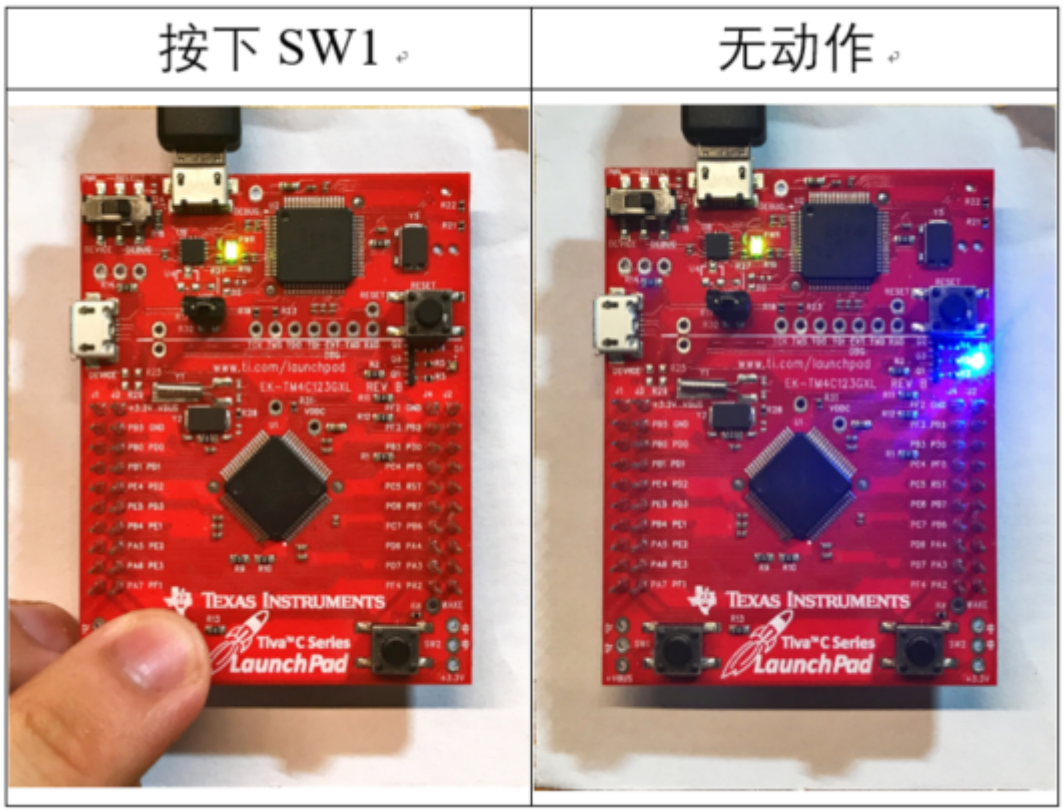
```
while(1){
    In = GPIO_PORTF_DATA_R&0x10;    // read PF4 into Sw1
    Out = In>>2;                    // shift into position PF2
    GPIO_PORTF_DATA_R = Out;        // output
}
```

while部分为对应的功能实现，主要做的事就是从Prot F 的DATA中读取数据，此处&0x10(0001 0000)也就是只读取PF4的数据。此处读出的数据只用两种情况0x10(0001 0000)或者0x00(0000 0000)此时在其右移两位得到两种情况0x08(0000 0100)或0x00(0000 0000)，其实就是将PF4的值赋值到PF2上，直接等同于控制蓝灯亮灭，此时根据之前实验原理中图我们可以知道开关没按下去为1按下去为0，也就是开关没按下去灯亮，按下去灯灭。‘

- **原代码流程**



- **原效果展示**

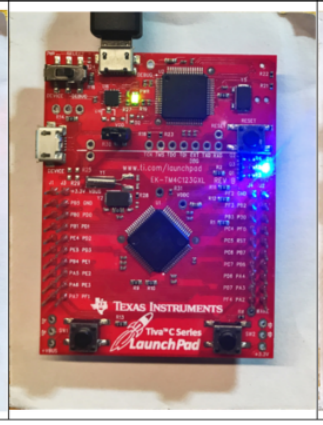


- **代码修改**

  此处根据我们上述分析，我们只要做的就是修改按钮和输出灯的位置。修改按钮我们只需要将In改成&0x01即可变为SW2作为输入。得到的是DATA的第一位，只要将其移动左移到2-4位即对应的PF1-3上即可改变灯亮。

```
while(1){
  In = GPIO_PORTF_DATA_R&0x01;  // read PF0 into Sw2
  //Out = In<<1;               // shift into position PF1 red
  //Out = In<<2;               // shift into position PF2 blue
  Out = In<<3;                 // shift into position PF3 green
  GPIO_PORTF_DATA_R = Out;     // output
}
```

- **修改效果**

按下 SW2	Out=In<<1	Out=In<<2	Out=In<<3
			

Lab2_2:定义位地址

- **代码理解**

```
#include <stdint.h>
#include "tm4c123gh6pm.h"

#define PF2   (*((volatile uint32_t *)0x40025010))
```

此处定义了PF2对应的地址

```
// Make PF2 an output, enable digital I/O, ensure alt. functions off
void SSR_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x20;        // 1) activate clock for Port F
  while((SYSCTL_PRGPIO_R&0x20)==0){}; // allow time for clock to start
                                    // 2) no need to unlock PF2
  GPIO_PORTF_PCTL_R &= ~0x00000F00; // 3) regular GPIO
  GPIO_PORTF_AMSEL_R &= ~0x04;      // 4) disable analog function on PF2
  GPIO_PORTF_DIR_R |= 0x04;         // 5) set direction to output
  GPIO_PORTF_AFSEL_R &= ~0x04;      // 6) regular port function
  GPIO_PORTF_DEN_R |= 0x04;         // 7) enable digital port
}
```

此处将进行了Prot F的初始化，首先还是激活F的时钟，因为此处只需要用到PF2所以不需要进行解锁。此处只设置PF2的GPIO Port Control，然后设定PF2为输出维，激活其数字I/O。

```
// Make PF2 high
void SSR_On(void){
  PF2 = 0x04;
//  GPIO_PORTF_DATA_R |= 0x04;
}
// Make PF2 low
void SSR_Off(void){
  PF2 = 0x00;
//  GPIO_PORTF_DATA_R &= ~0x04;
}
```

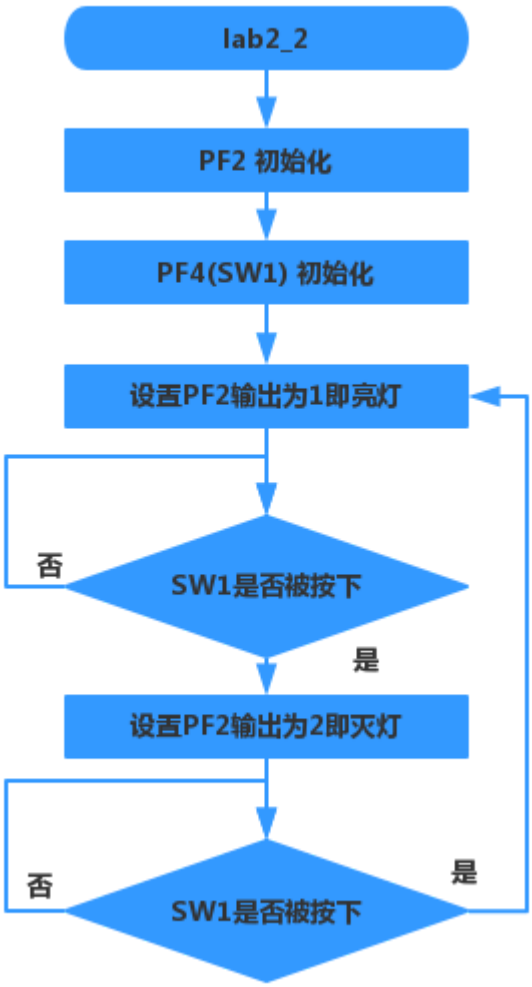
以上是控制灯亮灭的函数，PF2对应的为蓝灯，所以是控制蓝灯的亮灭，蓝灯亮时PF2=0x04，等同于将Port F的DATA第3位置1，灭灯时PF2=0x00，等同于将Port F 的DATA第3位置0。

```
int main(void){
  SSR_Init();               // initialize PF2 and make it output
  GPIO_PORTF_DIR_R &= ~0x10;// make PF4 in (PF4 built-in button #1)
  GPIO_PORTF_AFSEL_R &= ~0x10;// disable alt funct on PF4
  GPIO_PORTF_PUR_R |= 0x10; // enable pull-up on PF4
  GPIO_PORTF_DEN_R |= 0x10; // enable digital I/O on PF4
                            // configure PF4 as GPIO
  GPIO_PORTF_PCTL_R = (GPIO_PORTF_PCTL_R&0xFFF0FFFF)+0x00000000;
  while(1){
    SSR_On();
    while(GPIO_PORTF_DATA_R&0x10);        // wait for button press
    while((GPIO_PORTF_DATA_R&0x10) == 0); // wait for button release
    SSR_Off();
    while(GPIO_PORTF_DATA_R&0x10);        // wait for button press
    while((GPIO_PORTF_DATA_R&0x10) == 0); // wait for button release
  }
}
```

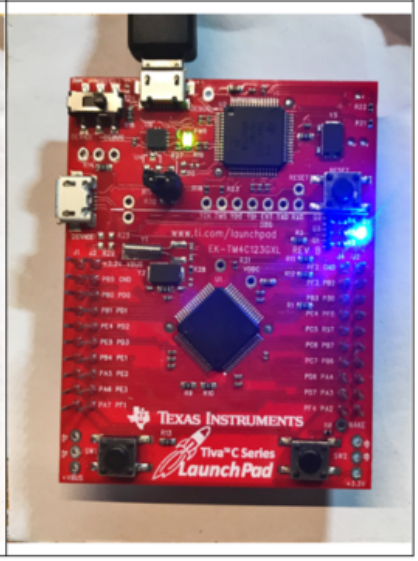
主函数中首先是进行了PF4的初始化，也就是SW1，设定其为输入口。

之后while循环中进行的是亮灯，然后等待SW1按下后释放，就关灯，然后SW1再次按下后释放就亮灯。也就是按一下SW1蓝灯灯就在亮灭之间变化。

- **原代码流程**



- **原代码效果**

无动作	按下 SW1	再次按下 SW1
		

- **代码修改**

此处我们将输入换为SW2，同时修改为控制三盏灯(也就是白灯)。

此处我们修改之前的宏定义改为PF3-1端口。

```
#define PF3_1   (*((volatile uint32_t *)0x40025038))
```

然后修改初始化端口部分

```
// Make PF3-1 an output, enable digital I/O, ensure alt. functions off
void SSR_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x20;        // 1) activate clock for Port F
  while((SYSCTL_PRGPIO_R&0x20)==0){}; // allow time for clock to start
                                    // 2) no need to unlock PF3-1
  GPIO_PORTF_PCTL_R &= ~0x0000FFF0; // 3) regular GPIO
  GPIO_PORTF_AMSEL_R &= ~0x0E;      // 4) disable analog function on PF3-1
  GPIO_PORTF_DIR_R |= 0x0E;         // 5) set direction to output
  GPIO_PORTF_AFSEL_R &= ~0x0E;      // 6) regular port function
  GPIO_PORTF_DEN_R |= 0x0E;         // 7) enable digital port
}
```

之后修改控制灯亮灭的函数改为3个LED同时亮灭

```
// Make PF3-1 high
void SSR_On(void){
  PF3_1 = 0x0E;
//  GPIO_PORTF_DATA_R |= 0x0E;
}
// Make PF2 low
void SSR_Off(void){
  PF3_1 = 0x00;
//  GPIO_PORTF_DATA_R &= ~0x0E;
}
```

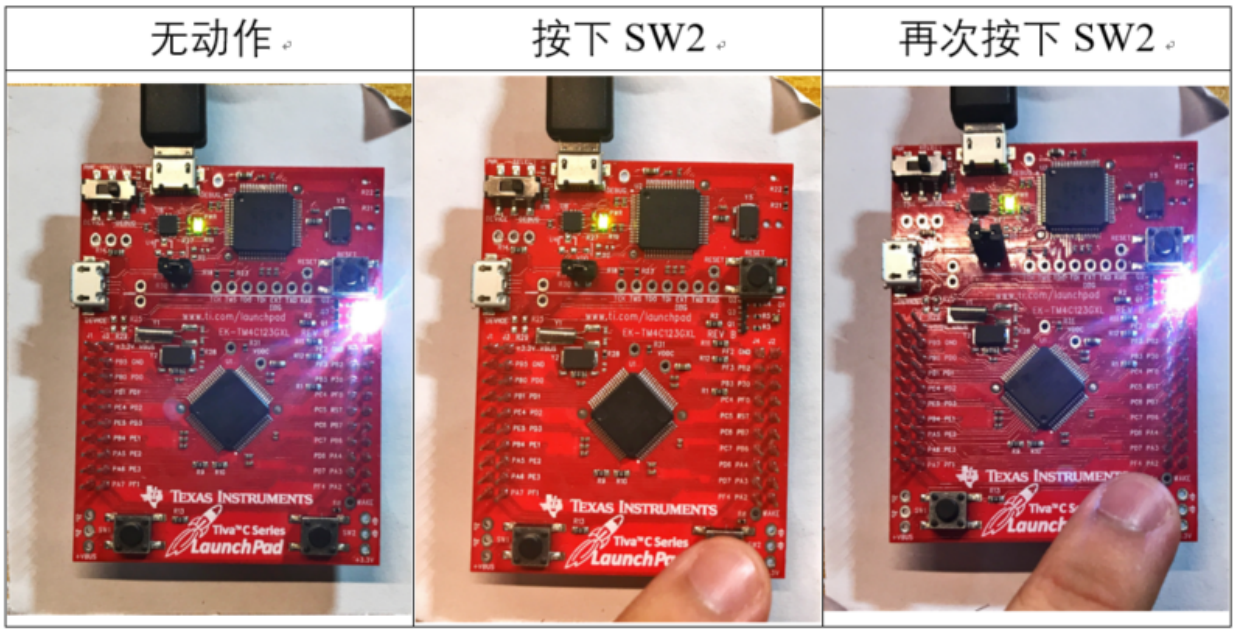
最后修改main函数中SW1的部分改成SW2，此处注意一点因为sw2对应的是PF0所以此时要增加对于PF0端口的解锁

```
int main(void){
  SSR_Init();               // initialize PF2 and make it output
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   //unlock GPIO Port F
  GPIO_PORTF_CR_R |= 0x01;              // allow changes to PF0
  GPIO_PORTF_DIR_R &= ~0x01;// make PF4 in (PF4 built-in button #1)
  GPIO_PORTF_AFSEL_R &= ~0x01;// disable alt funct on PF4
  GPIO_PORTF_PUR_R |= 0x01; // enable pull-up on PF4
  GPIO_PORTF_DEN_R |= 0x01; // enable digital I/O on PF4
                            // configure PF4 as GPIO
  GPIO_PORTF_PCTL_R = (GPIO_PORTF_PCTL_R&0xFFFFFFF0)+0x00000000;
  while(1){
        SSR_On();
    while(GPIO_PORTF_DATA_R&0x01);        // wait for button press
    while((GPIO_PORTF_DATA_R&0x01) == 0); // wait for button release
```

```
        SSR_Off();
    while(GPIO_PORTF_DATA_R&0x01);        // wait for button press
    while((GPIO_PORTF_DATA_R&0x01) == 0); // wait for button release
  }
}
```

修改完成后效果应该是按一下白灯灭在按一下亮起。

- **修改后效果**



Lab2_3: Debug mode-select “Use simulator”

- **代码理解**

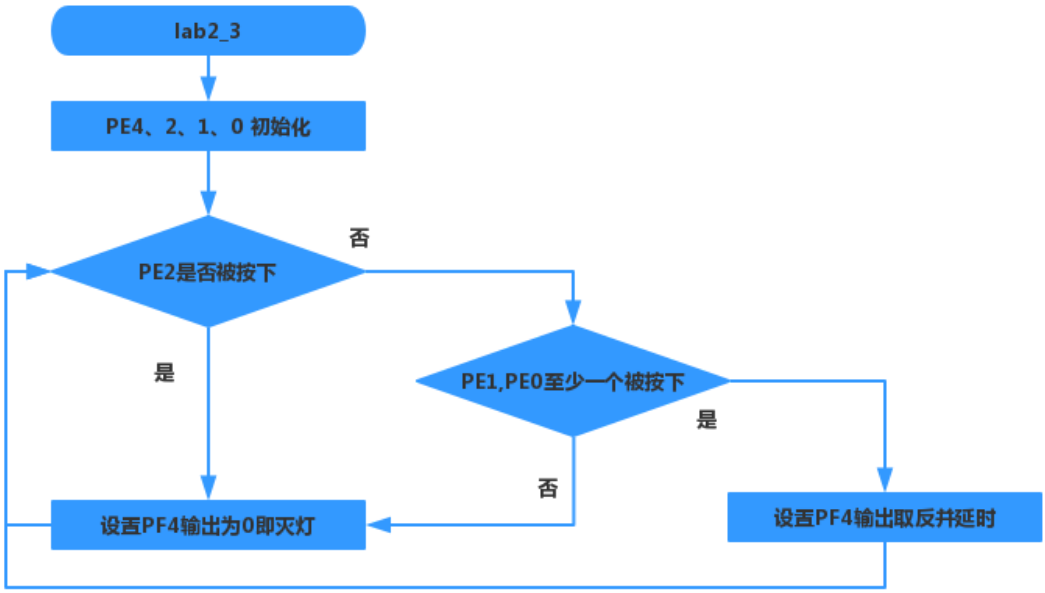
```
SYSCTL_RCGC2_R |= 0x10; // Port E clock
delay = SYSCTL_RCGC2_R;// wait 3-5 bus cycles
GPIO_PORTE_DIR_R |= 0x10;// PE4 output
GPIO_PORTE_DIR_R &= ~0x07;// PE2,1,0 input
GPIO_PORTE_AFSEL_R &= ~0x17; // not alternative
GPIO_PORTE_AMSEL_R &= ~0x17;// no analog
GPIO_PORTE_PCTL_R &= ~0x000F0FFF; // bits for PE4,PE2,PE1,PE0
GPIO_PORTE_DEN_R |= 0x17;// enable PE4,PE2,PE1,PE0
```

此处是进行端口初始化同样激活Port E的时钟，然后设定PE4为输出，PE2-0位输入。

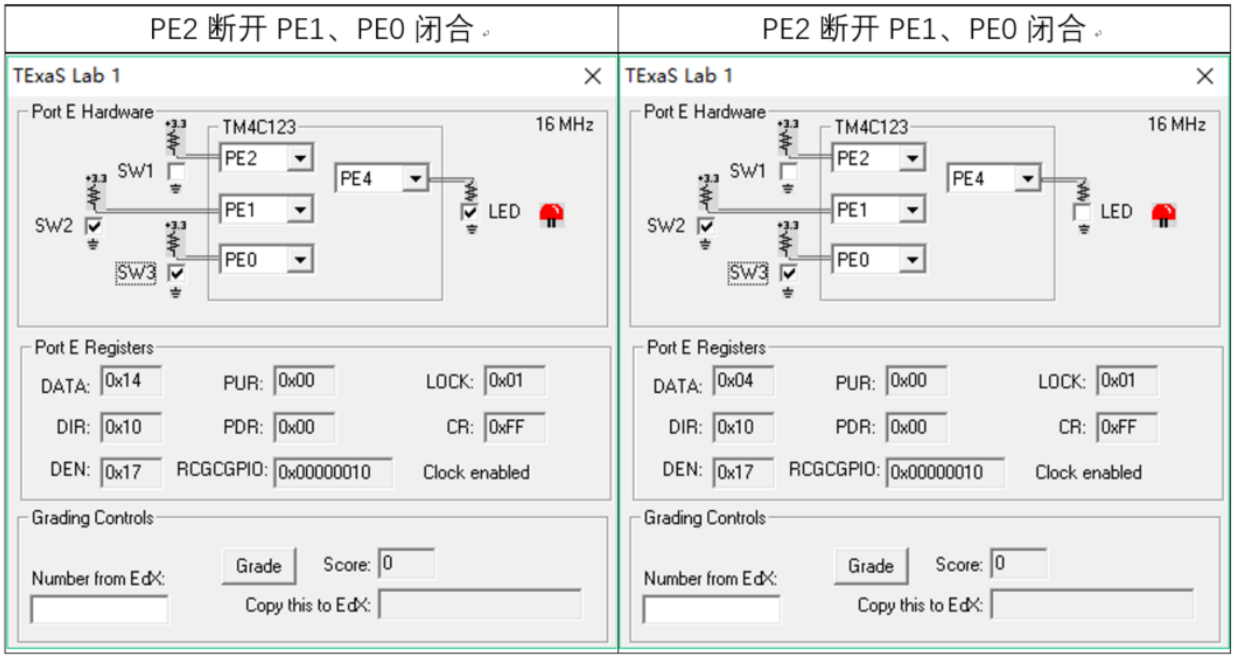
```
while(1){
    arm = GPIO_PORTE_DATA_R&0x04; // arm 0 if deactivated, 1 if activated(PE2)
    sensor = GPIO_PORTE_DATA_R&0x03; // 1 means ok, 0 means break in
    if((arm==0x04)&&(sensor != 0x03)){
        GPIO_PORTE_DATA_R ^= 0x10; // toggle output for alarm
        delay=100;  // 100ms delay makes a 5Hz period
    }else{
        GPIO_PORTE_DATA_R &= ~0x10; // LED off if deactivated
    }
 }
```

此处先取了Port E 的第3位的数据也就是PE2放在arm中，然后取了Port E DATA的第1、2位数据放在sensor中。此时如果arm=0x04也就是PortE 第3位为1(PE2 对应开关断开)同时sensor!=0x03也就是PortE的1、2位不同时为1(PE1,PE0对应的开关不同时断开，即至少有一个闭合)，此时PE4的输出将会在每隔100ms变化一次，也就是对应的LED灯亮灭变化；否则LED灯不亮。

- **代码流程**

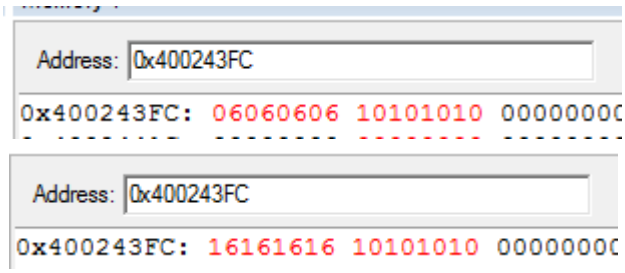


- **代码结果**



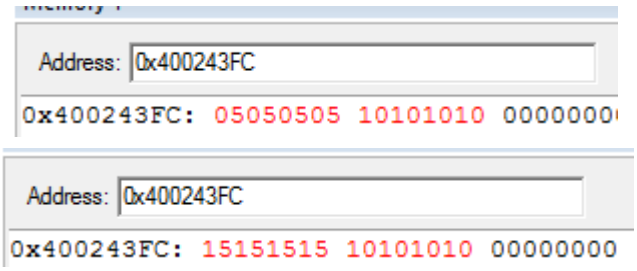
- 当我们按下PE0之后我们可以看到此时PE4在跳变

从寄存器0x400243FC(Port E的状态)代表PE4-0的值的和(0x02+0x04+0x10=0x16)或(0x02+0x04+0x00=0x06),

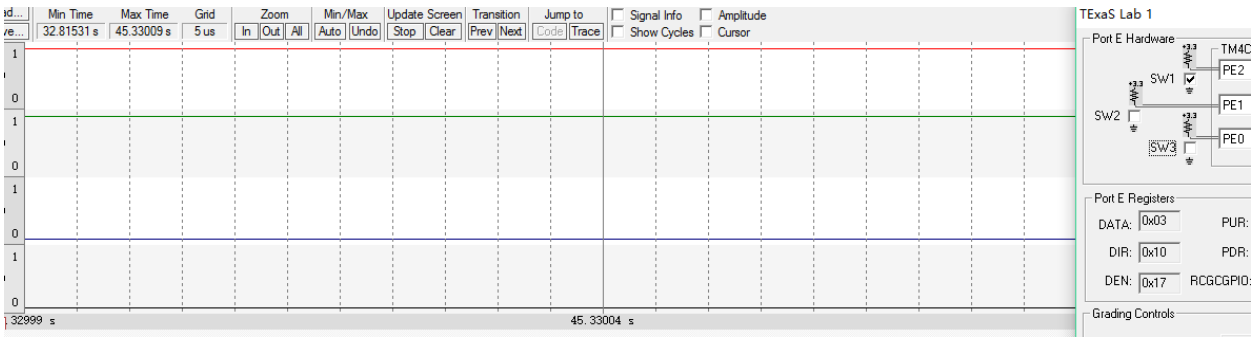


- 当我们按下PE1之后我们可以看到此时PE4在跳变

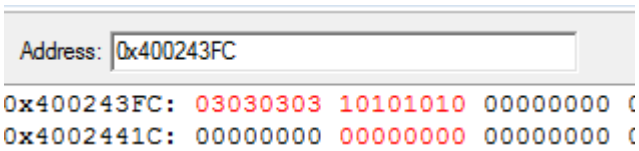
从寄存器0x400243FC(Port E的状态)代表PE4-0的值的和(0x01+0x04+0x10=0x15)或(0x01+0x04+0x00=0x05)，



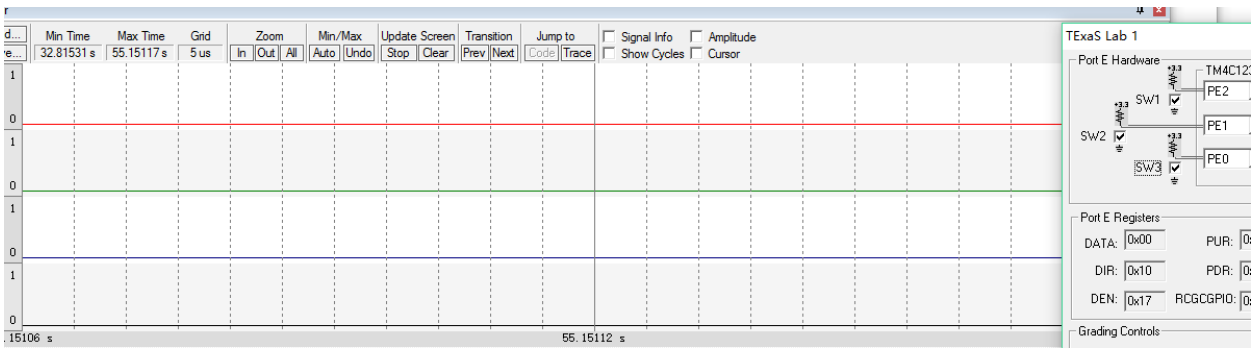
- 如果当我们按下PE2之后



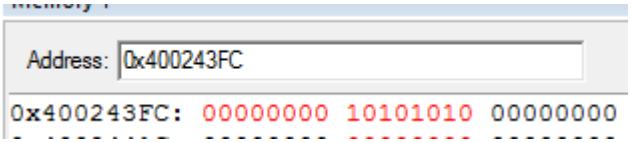
此时符合之前理论中不会改变LED灯的情况此时从寄存器0x400243FC的值恒为(0x01+0x02=0x03)



- 如果当我们按下PE2之后在按下PE1和PE0



此时符合之前理论中不会改变LED灯的情况，此时从寄存器0x400243FC的值恒为0因为全部开关按下。



其他情况不在展示

- **结论:**这段代码就是相当于一个报警系统，当PE2断开时报警系统启动，如果此时有任意一个sensor触发(PE1或者PE0按下)那么就会使得警报触发(PE4的led亮灭)，如果PE2按下此时警报系统关闭，任何一个sensor都不会触发警报

5. 实验心得

这次实验只要是学会使用调试模式进行对代码的调试，从而了解代码在不同状态下所做的事情，通过值的变化我们可以很清楚的知道这个时间点之后发生了什么。调试也可以验证我们之前对于代码的理论上的理解，并给与检验。通过这次实验对于调试方式有了很好的掌握。