EE226 Lab3

Led Lighting With Customized Scenario

MICROPROCESSORS LABORATORY

Ömer Karslıoğlu

Introduction

First of all, I wrote the codes to intialize port and some peripherals. Later, I made a detailed research on general purpose input / output. It would be more useful to explain my work over my code that I have prepared.I would like to explain the initialize operations before explaining the four experiments.

Libraries

```
1  #include <stdint.h>
2  #include <stdlib.h>
```

The <stdint.h> header shall declare sets of integer types having specified widths, and shall define corresponding sets of macros. It shall also define macros that specify limits of integer types corresponding to types defined in other standard headers.
The <stdlib.h> header defines four variable types, several macros, and various functions for performing general functions.

Address And Some Parameters Definitions

```
5  #define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE608))
6  #define SYSCTL_RCGC2_GPIOE      0x00000010   // Port E Clock Gating Control
```

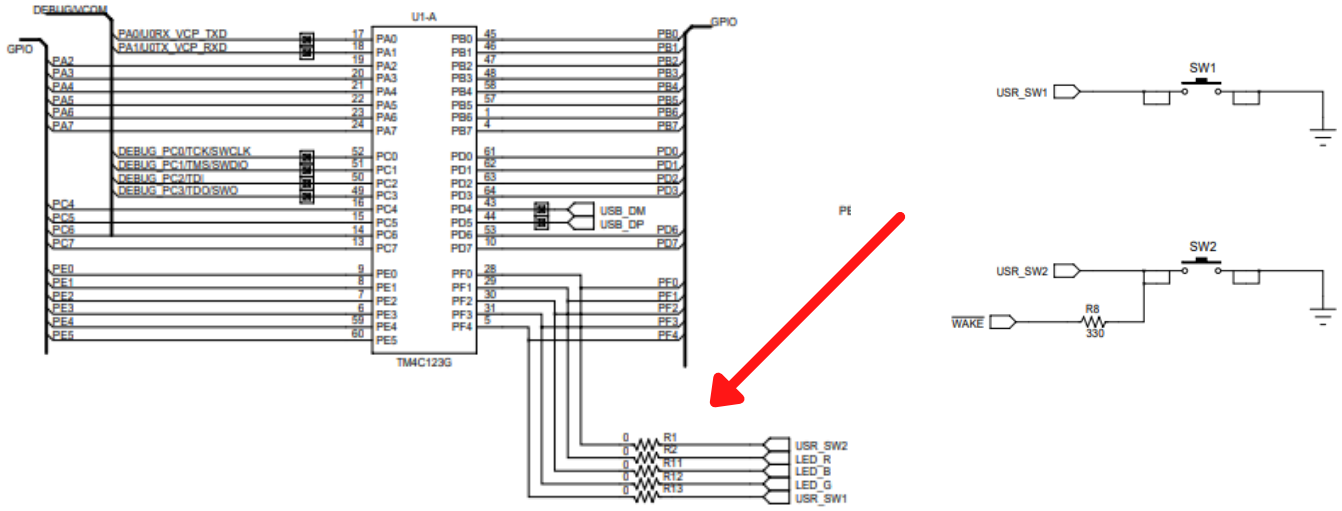
On the 5th line, we defined the address of rcg peripheral for your clock configuration with the help of pointers. We will use the number 0x20 to intialize the PORTF.

```
10  #define GPIO_PORTE_LOCK_R       (*((volatile unsigned long *)0x40024520))
11  #define GPIO_PORTE_CR_R         (*((volatile unsigned long *)0x40024524))
12
13  #define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
14  #define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
15  #define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
```

The GPIOLOCK register must be enabled write access to the GPIOCR register . GPIOLOCK register should be (from datasheet on 684) offset 0x520 .Commit register should be (from datasheet on 685) offset 0x524 .
Although I don't need to initiliaze AMSEL register , PCTL register and AFSEL register for this application , I set them all.

```
17  #define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
18
19  #define GPIO_PORTE_DATA_R       (*((volatile unsigned long *)0x400243FC))
20  #define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
21  #define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
```

To configure the PORTF, I need to define addresses the DATA register, the DIR (direction to set input / output) register and enable register (DEN) .
In addition to this , I set pull-up register for inputs .
While defining them, I used datasheet, especially memory mapping.



The image from tiva c4 series user's manual from page 20 . (Schematic Image)

As you see ,user switch 1 is connected to PF4. The user switch 2 is connected to PF0, so I defined sw1 as 0x10 (0b00010000) and sw2 as 0x01 (0b00000001).

Functions

Firstly , I defined some functions for each part and experiment . One of them is delay function . Now I wanna mention about initializing function .(I gonna explain others .)

```
49  void initialize_PortF(void)
50  {
51    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
52    while(!( SYSCTL_PRGPIO_R& 0x20)){} //
53
54    GPIO_PORTF_LOCK_R = 0x4C4F434B;    //
55    GPIO_PORTF_CR_R = 0x1F;            //
56                                       //
57    GPIO_PORTF_DIR_R = 0x0E; //01110 IOOO
58    GPIO_PORTF_PUR_R = 0x11;           //
59    GPIO_PORTF_DEN_R = 0x1F; //0011111 PE
60
61    GPIO_PORTF_DATA_R = 0x00; // reset to
62  }
```

I set the clock peripheral to equal 0x20 for portf. I set the lock register as unlock and set the commit register for inputs.
For Direction register I need to give the following information:
0 means to set as input .
1 means to set as output .
Therefore , I set as 0x01110 (0x0E) .
I set the pull-up register to 0x01 (PF0) and 0X10 (PF4).[0x01 + 0x10 = 0x11] .
Then, I enabled inputs and outputs.

```
void initialize_PortE(void)
{
  SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOE;
  while(!( SYSCTL_PRGPIO_R& 0x10)){ } //waiting to enable clk

  GPIO_PORTE_LOCK_R  = 0x4C4F434B;   // unlock GPIO Port E
  GPIO_PORTE_CR_R    = 0x1F;           // allow changes to PE-0
                                     // only PF0 needs to be unlocked, other bits can't be locked
  GPIO_PORTE_DIR_R   = 0x0F; //01111 IOOOO
  //GPIO_PORTE_PUR_R   = 0x00;          // NOT enable pull-up on PF0 and PF4 (RXTERNAL PULL UP)
  GPIO_PORTE_DEN_R   = 0x1F; //0011111  PE1 , PE2 , PE3 , PE4  ACTIVE

  //GPIO_PORTE_DATA_R  = 0x00; // reset to initialize all pins

  PE_1_2_3=0x00; //specified address
}
```

First Experiment

```
int thePartOneOfDutyCycle=20; // %20 percent of the dutycycle
int thePartTwoOfDutyCycle=80; // %80 percent of the dutycycle
31  int main(void)
32  {
33    initialize_PortF();
34
35    while(1)
36    {
37      GPIO_PORTF_DATA_R |= 0x02; // red (pf1) 10
38      // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
39      delay(125*thePartOneOfDutyCycle);
40      // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
41      GPIO_PORTF_DATA_R ^= 0x02; // toggle
42      delay(125*thePartTwoOfDutyCycle);
43    }
44
45
46    return 0;
47  }
```

After initializing the portf, I light the led in 20% of 8 Hz with the delay function that I prepared.To understand this, it is first necessary to understand the delay function.

```
65  //Our mcu is working on 80 MHz
66  // Specified Delay Function
67  // @Def :
68  // 1 ms delay nearly
69  // if sec equals to 10 this means 1 duty cycle
70  void delay(int sec)
71  {
72    int x=80; //x=8000;//I delete two zeros for specified process
73    while(sec>0)
74    {
75      sec--;
76      while(x>0)
77      {
78        x--;
79      }
80    }
81  }
```

MCU works at 80 MHz. I have prepared a delay function that will increase by one millisecond. Later, I deleted two zeros here to get rid of the floating numbers .

Notification : T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms

First Experiment Codes

```
#include <stdint.h>
#include <stdlib.h>
//#include <tm4c123gh6pm.h> // I can use this library or i can write the definition of addresses :

#define SYSCTL_RCGC2_R (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGC2_GPIOF 0x00000020 // Port F Clock Gating Control

#define SYSCTL_PRGPIO_R        (*((volatile unsigned long *)0x400FEA08))

#define GPIO_PORTF_LOCK_R      (*((volatile unsigned long *)0x40025520)) // The GPIOLOCK register
//enables write access to the GPIOCR register  (datasheet on 684) offset 0x520
#define GPIO_PORTF_CR_R        (*((volatile unsigned long *)0x40025524)) // commit register //(datasheet
//on 685) offset 0x524

#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))

#define GPIO_PORTF_PUR_R       (*((volatile unsigned long *)0x40025510)) // pull-up (datasheet on //677)
//offset 0x510

#define GPIO_PORTF_DATA_R (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R (*((volatile unsigned long *)0x4002551C))

#define PF1                    (*((volatile unsigned long *)0x40025008))  // pin specifying address

void delay(int sec);
void initialize_PortF(void);

int thePartOneOfDutyCycle=20; // %20 percent of the dutycycle
int thePartTwoOfDutyCycle=80; // %80 percent of the dutycycle
```

```
int main(void)
{
initialize_PortF();

while(1)
{
GPIO_PORTF_DATA_R |= 0x02; // red (pf1) 10
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
delay(125*thePartOneOfDutyCycle);
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
GPIO_PORTF_DATA_R ^= 0x02; // toggle
delay(125*thePartTwoOfDutyCycle);
}


return 0;
}

void initialize_PortF(void)
{
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
while(!( SYSCTL_PRGPIO_R& 0x20)){} //waiting to enable clk

GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x1F;           // allow changes to PF4-0
// only PF0 needs to be unlocked, other bits can't be locked
GPIO_PORTF_DIR_R = 0x0E; //01110 IOOOI
  GPIO_PORTF_PUR_R = 0x11;          // enable pull-up on PF0 and PF4
GPIO_PORTF_DEN_R = 0x1F; //0011111 PE0 (SW2) , PE1 , PE2 , PE3 , PE4 (SW1) ACTIVE

GPIO_PORTF_DATA_R = 0x00; // reset to initialize all pins
}


//Our mcu is working on 80 MHz
// Specified Delay Function
// @Def :
// 1 ms delay nearly
// if sec equals to 10 this means 1 duty cycle
void delay(int sec)
{
int x=80; //x=8000;//I delete two zeros for specified process
while(sec>0)
{
sec--;
while(x>0)
{
x--;
}
}
}
```

Second Experiment

```
32  int main(void)
33  {
34    initialize_PortF();
35
36    thePartOneOfDutyCycle=1;
37    thePartTwoOfDutyCycle=99;
38    dutyCycle = 750;
40    while(1)
41    {
42      for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
43      {
44        thePartTwoOfDutyCycle--;
45        PF1 |= 0x02; // red (pf1)
46        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
47        delay(dutyCycle*thePartOneOfDutyCycle);
48        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
49        PF1 ^= 0x02; // toggle
50        delay(dutyCycle*thePartTwoOfDutyCycle);
51      }
52      for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
53      {
54        thePartOneOfDutyCycle--;
55        PF1 |= 0x02; // red (pf1)
56        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
57        delay(dutyCycle*thePartOneOfDutyCycle);
58        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
59        PF1 ^= 0x02; // toggle
60        delay(dutyCycle*thePartTwoOfDutyCycle);
61      }
62    }
63    return 0;
64  }
```

In two for loops, I performed the breathing led application in 200 steps in total, with approximately 100 brightness increase and 100 brightness decrease. In addition, I expanded the dutycycle to make it look better on the eye. (The manual says duty cycle can be changed for the second experiment). If requested, "dutyCycle = 125;" 8 Hz can be made.

```
22
23  #define PF1                     (*((volatile unsigned long *)0x40025008))
24
```

By the way, here I have defined an address where only pf1 can be changed by defining a private address. When using this address, nothing can be written to other places of the data register.

Second Experiment Codes

```
#include <stdint.h>
#include <stdlib.h>
//#include <tm4c123gh6pm.h> // I can use this library or i can write the definition of addresses :

#define SYSCTL_RCGC2_R (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGC2_GPIOF 0x00000020 // Port F Clock Gating Control

#define SYSCTL_PRGPIO_R        (*((volatile unsigned long *)0x400FEA08))

#define GPIO_PORTF_LOCK_R      (*((volatile unsigned long *)0x40025520)) // The GPIOLOCK register
//enables write access to the GPIOCR register  (datasheet on 684) offset 0x520
#define GPIO_PORTF_CR_R        (*((volatile unsigned long *)0x40025524)) // commit register (datasheet on
//685) offset 0x524

#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))

#define GPIO_PORTF_PUR_R       (*((volatile unsigned long *)0x40025510)) // pull-up (datasheet on 677)
//offset 0x510

#define GPIO_PORTF_DATA_R (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R (*((volatile unsigned long *)0x4002551C))

#define PF1                    (*((volatile unsigned long *)0x40025008))  // pin specified address

void delay(int sec);
void initialize_PortF(void);

int thePartOneOfDutyCycle; //yüzdelik dilim
int thePartTwoOfDutyCycle;
int dutyCycle;
```

```
int main(void)
{
initialize_PortF();

thePartOneOfDutyCycle=1;
thePartTwoOfDutyCycle=99;
dutyCycle = 750;

while(1)
{
for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
{
thePartTwoOfDutyCycle--;
PF1 |= 0x02; // red (pf1)
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
delay(dutyCycle*thePartOneOfDutyCycle);
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
PF1 ^= 0x02; // toggle
delay(dutyCycle*thePartTwoOfDutyCycle);
}
for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
{
thePartOneOfDutyCycle--;
PF1 |= 0x02; // red (pf1)
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
delay(dutyCycle*thePartOneOfDutyCycle);
// T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
PF1 ^= 0x02; // toggle
delay(dutyCycle*thePartTwoOfDutyCycle);
}
}
return 0;
}

void initialize_PortF(void)
{
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
while(!( SYSCTL_PRGPIO_R& 0x20)){ } //waiting to enable clk

GPIO_PORTF_LOCK_R  = 0x4C4F434B;   // unlock GPIO Port F
  GPIO_PORTF_CR_R    = 0x1F;           // allow changes to PF4-0
// only PF0 needs to be unlocked, other bits can't be locked
GPIO_PORTF_DIR_R   = 0x0E; //01110 IOOOI
  GPIO_PORTF_PUR_R   = 0x11;          // enable pull-up on PF0 and PF4
GPIO_PORTF_DEN_R   = 0x1F; //0011111 PE0 (SW2) , PE1 , PE2 , PE3 , PE4 (SW1) ACTIVE

GPIO_PORTF_DATA_R  = 0x00; // reset to initialize all pins

PF1=0x00; //specified address
}
```

```
//Our mcu is working on 80 MHz
// Specified Delay Function
// @Def :
// 1 ms delay nearly
// if sec equals to 10 this means 1 duty cycle
void delay(int sec)
{
int x=80; //x=(8000);// I delete two zeros for specify process
while(sec>0)
{
 sec--;
 while(x>0)
 {
 x--;
 }
}
}
```

Third Experiment

```
98  //@Def : to read User User_Switch1
99  //gives value when switch1 is pressed
100 unsigned long readSwitch1()
101 {
102   return (PF4&0x10); // 0 b 0001 0000
103 }
104
```

I have defined a function that reads the value that PF4 takes.

```
34  int main(void)
35  {
36    initialize_PortF();
37
38    thePartOneOfDutyCycle=1;
39    thePartTwoOfDutyCycle=99;
40    dutyCycle = 125;
41
42    while(1)
43    {
44      if(!(readSwitch1())) //active low switch , it's mean is that switch is cconnected to groud with pull up res.
45      {
46          PF1 |= 0x02; // red (pf1)
47          // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
48          delay(dutyCycle*20);
49          // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
50          PF1 ^= 0x02; // toggle
51          delay(dutyCycle*80);
52      }
53
```

```
54        else
55        {
56          for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
57          {
58            thePartTwoOfDutyCycle--;
59            PF1 |= 0x02; // red (pf1)
60            // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
61            delay(dutyCycle*thePartOneOfDutyCycle);
62            // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
63            PF1 ^= 0x02; // toggle
64            delay(dutyCycle*thePartTwoOfDutyCycle);
65          }
66          for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
67          {
68            thePartOneOfDutyCycle--;
69            PF1 |= 0x02; // red (pf1)
70            // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
71            delay(dutyCycle*thePartOneOfDutyCycle);
72            // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
73            PF1 ^= 0x02; // toggle
74            delay(dutyCycle*thePartTwoOfDutyCycle);
75          }
76        }
77    }
78    return 0;
79  }
```

Combining experiment 1 and experiment 2, I reused the same codes here.
There are two state . When SW1 is on, 8Hz LED blinking will be on. When SW1 is off, LED breathing will be on.

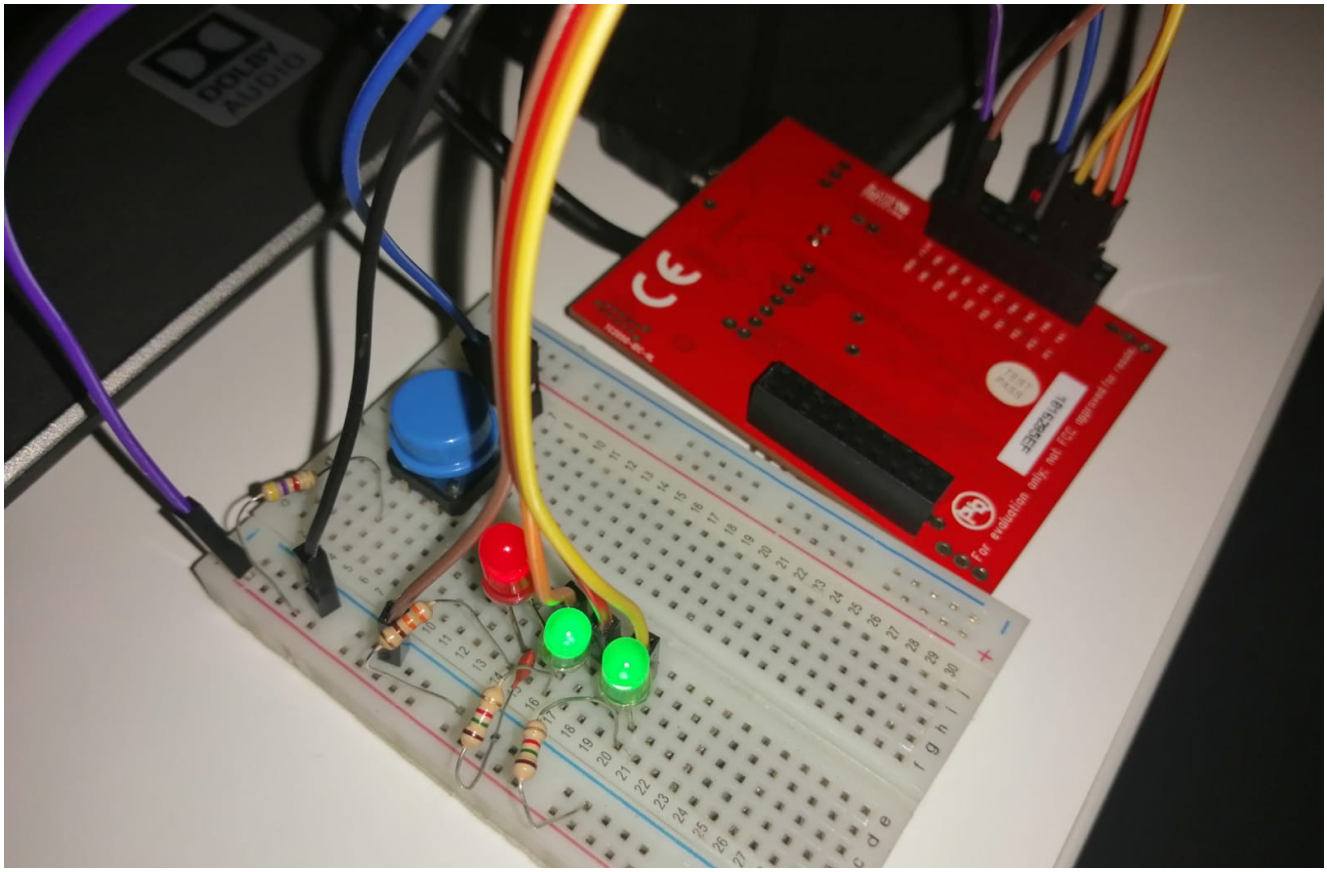
Third Experiment Codes

```
#include <stdint.h>
#include <stdlib.h>
//#include <tm4c123gh6pm.h> // I can use this library or i can write the definition of addresses :

#define SYSCTL_RCGC2_R (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGC2_GPIOF 0x00000020 // Port F Clock Gating Control

#define SYSCTL_PRGPIO_R        (*((volatile unsigned long *)0x400FEA08))

#define GPIO_PORTF_LOCK_R      (*((volatile unsigned long *)0x40025520)) // The GPIOLOCK register
//enables write access to the GPIOCR register  (datasheet on 684) offset 0x520
#define GPIO_PORTF_CR_R        (*((volatile unsigned long *)0x40025524)) // commit register (datasheet on
//685) offset 0x524

#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))

#define GPIO_PORTF_PUR_R       (*((volatile unsigned long *)0x40025510)) // pull-up (datasheet on 677)
//offset 0x510

#define GPIO_PORTF_DATA_R (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R (*((volatile unsigned long *)0x4002551C))

#define PF1                    (*((volatile unsigned long *)0x40025008))  // pin specified address (RED LED)
#define PF4 (*((volatile unsigned long *)0x40025040))  // pin specified address  (USER SWITCH1)

void delay(int sec);
void initialize_PortF(void);
unsigned long readSwitch1();

int thePartOneOfDutyCycle; //yüzdelik dilim
int thePartTwoOfDutyCycle;
int dutyCycle;
```

```
int main(void)
{
initialize_PortF();

thePartOneOfDutyCycle=1;
thePartTwoOfDutyCycle=99;
dutyCycle = 125;

while(1)
{
 if(!(readSwitch1())) //active low switch , it's mean is that switch is cconnected to groud with pull up res.
 {
 PF1 |= 0x02; // red (pf1)
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*20);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PF1 ^= 0x02; // toggle
 delay(dutyCycle*80);
 }

 else
 {
 for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
 {
 thePartTwoOfDutyCycle--;
 PF1 |= 0x02; // red (pf1)
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*thePartOneOfDutyCycle);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PF1 ^= 0x02; // toggle
 delay(dutyCycle*thePartTwoOfDutyCycle);
 }
 for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
 {
 thePartOneOfDutyCycle--;
 PF1 |= 0x02; // red (pf1)
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*thePartOneOfDutyCycle);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PF1 ^= 0x02; // toggle
 delay(dutyCycle*thePartTwoOfDutyCycle);
 }
 }
}
return 0;
}
```

```
void initialize_PortF(void)
{
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
while(!( SYSCTL_PRGPIO_R& 0x20)){ } //waiting to enable clk

GPIO_PORTF_LOCK_R = 0x4C4F434B; // unlock GPIO Port F
 GPIO_PORTF_CR_R = 0x1F; // allow changes to PF4-0
 // only PF0 needs to be unlocked, other bits can't be locked
GPIO_PORTF_DIR_R = 0x0E; //01110 IOOOI
 GPIO_PORTF_PUR_R = 0x11; // enable pull-up on PF0 and PF4
GPIO_PORTF_DEN_R = 0x1F; //0011111 PE0 (SW2) , PE1 , PE2 , PE3 , PE4 (SW1) ACTIVE

GPIO_PORTF_DATA_R = 0x00; // reset to initialize all pins

PF1=0x00; //specified address
}

//@Def : to read User User_Switch1
//gives value when switch1 is pressed
unsigned long readSwitch1()
{
return (PF4&0x10); // 0 b 0001 0000
}

//Our mcu is working on 80 MHz
// Specified Delay Function
// @Def :
// 1 ms delay function nearly
// if sec equals to 10 this means 1 duty cycle
void delay(int sec)
{
int x=80; //x=(8000);// I delete two zeros for specify process
while(sec>0)
{
 sec--;
 while(x>0)
 {
 x--;
 }
}
}
```

Fourth Experiment



I wrote the version of the code we wrote in experiment 3 that can be applied to the external circuit in experiment 4.I set all the necessary configurations for the PORTE.

Let me briefly summarize the first lead external circuit:

PE1, PE2 and PE3 pins are connected to the LEDs. (PE1 and PE2 are connected to the green led, PE3 to the red led.). I connected the leds with the protection resistor to the gnd line. I connected the switch to PE4 and added a 5KOhm pull-up resistor. I inactivated the internal pull-up in the configuration settings.

```
#define PE_1_2_3                (*((volatile unsigned long *)0x40024038))  // pin specified address (LEDS)
#define PE4                     (*((volatile unsigned long *)0x40024040))  // pin specified address  (SWITCH)

void delay(int sec);
void initialize_PortE(void);
unsigned long readSwitch1();

int thePartOneOfDutyCycle; //yüzdelik dilim
int thePartTwoOfDutyCycle;
int dutyCycle;
```

First of all, I made my specified address definitions, function definitions and variable definitions.

```
void initialize_PortE(void)
{
  SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOE;
  while(!( SYSCTL_PRGPIO_R& 0x10)){ } //waiting to enable clk

  GPIO_PORTE_LOCK_R   = 0x4C4F434B;   // unlock GPIO Port E
  GPIO_PORTE_CR_R     = 0x1F;           // allow changes to PE-0
                                      // only PF0 needs to be unlocked, other bits can't be locked
  GPIO_PORTE_DIR_R    = 0x0F; //01111 IOOOO
  //GPIO_PORTE_PUR_R    = 0x00;          // NOT enable pull-up on PF0 and PF4 (RXTERNAL PULL UP)
  GPIO_PORTE_DEN_R    = 0x1F; //0011111  PE1 , PE2 , PE3 , PE4  ACTIVE

  //GPIO_PORTE_DATA_R  = 0x00; // reset to initialize all pins

  PE_1_2_3=0x00; //specified address
}
```

I wrote initialization function for PORTE.

```
//@Def : to read User User_Switch1
//gives value when switch1 is pressed
unsigned long readSwitch1()
{
  return (PE4&0x10); // 0 b 0001 0000
}

//Our mcu is working on 80 MHz
// Specified Delay Function
// @Def :
// 1 ms delay function nearly
// if sec equals to 10 this means 1 duty cycle
void delay(int sec)
{
  int x=80; //x=(8000);// I delete two zeros for specify process
  while(sec>0)
  {
    sec--;
    while(x>0)
    {
      x--;
    }
  }
}
```

After preparing my switch reading and delay functions, I started writing my main code.

```
int main(void)
{
  initialize_PortE();

  thePartOneOfDutyCycle=1;
  thePartTwoOfDutyCycle=99;
  dutyCycle = 125;

  while(1)
  {
    if(!(readSwitch1())) //active low switch , it's mean is that switch is cconnected to groud with pull up res.
    {
        PE_1_2_3 |= 0x0F; // LIGHT LED
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        delay(dutyCycle*20);
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        PE_1_2_3 ^= 0x0F; // toggle
        delay(dutyCycle*80);
    }
```

```
    else
    {
      for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
      {
        thePartTwoOfDutyCycle--;
        PE_1_2_3 |= 0x0F; // LIGHT LED
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        delay(dutyCycle*thePartOneOfDutyCycle);
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        PE_1_2_3 ^= 0x0F; // toggle
        delay(dutyCycle*thePartTwoOfDutyCycle);
      }
      for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
      {
        thePartOneOfDutyCycle--;
        PE_1_2_3 |= 0x0F; // LIGHT LED
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        delay(dutyCycle*thePartOneOfDutyCycle);
        // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
        PE_1_2_3 ^= 0x0F; // toggle
        delay(dutyCycle*thePartTwoOfDutyCycle);
      }
    }
  }
  return 0;
}
```

There are two state again. When switch is pressed, 8Hz LED blinking will be on. When switch is off, LED breathing is on. Here again, I remind you that three LEDs do the same task.

Fourth Experiment Codes

```
#include <stdint.h>
#include <stdlib.h>
//#include <tm4c123gh6pm.h> // I can use this library or i can write the definition of addresses :

#define SYSCTL_RCGC2_R (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGC2_GPIOE 0x00000010 // Port E Clock Gating Control

#define SYSCTL_PRGPIO_R        (*((volatile unsigned long *)0x400FEA08))

#define GPIO_PORTE_LOCK_R      (*((volatile unsigned long *)0x40024520)) // The GPIOLOCK register
//enables write access to the GPIOCR register  (datasheet on 684) offset 0x520
#define GPIO_PORTE_CR_R        (*((volatile unsigned long *)0x40024524)) // commit register (datasheet on
//685) offset 0x524

#define GPIO_PORTE_AMSEL_R     (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R      (*((volatile unsigned long *)0x4002452C))
#define GPIO_PORTE_AFSEL_R     (*((volatile unsigned long *)0x40024420))

#define GPIO_PORTE_PUR_R       (*((volatile unsigned long *)0x40024510)) // pull-up (datasheet on 677)
//offset 0x510

#define GPIO_PORTE_DATA_R (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_DEN_R (*((volatile unsigned long *)0x4002451C))

#define PE_1_2_3               (*((volatile unsigned long *)0x40024038))  // pin specified address (LEDS)
#define PE4 (*((volatile unsigned long *)0x40024040))  // pin specified address  (SWITCH)

void delay(int sec);
void initialize_PortE(void);
unsigned long readSwitch1();

int thePartOneOfDutyCycle; //yüzdelik dilim
int thePartTwoOfDutyCycle;
int dutyCycle;
```

```
int main(void)
{
initialize_PortE();

thePartOneOfDutyCycle=1;
thePartTwoOfDutyCycle=99;
dutyCycle = 125;

while(1)
{
 if(!(readSwitch1())) //active low switch , it's mean is that switch is cconnected to groud with pull up res.
 {
 PE_1_2_3 |= 0x0F; // LIGHT LED
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*20);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PE_1_2_3 ^= 0x0F; // toggle
 delay(dutyCycle*80);
 }

 else
 {
 for(;thePartOneOfDutyCycle<=99;thePartOneOfDutyCycle++)
 {
 thePartTwoOfDutyCycle--;
 PE_1_2_3 |= 0x0F; // LIGHT LED
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*thePartOneOfDutyCycle);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PE_1_2_3 ^= 0x0F; // toggle
 delay(dutyCycle*thePartTwoOfDutyCycle);
 }
 for(;thePartTwoOfDutyCycle<=99;thePartTwoOfDutyCycle++)
 {
 thePartOneOfDutyCycle--;
 PE_1_2_3 |= 0x0F; // LIGHT LED
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 delay(dutyCycle*thePartOneOfDutyCycle);
 // T=1/f , so 1/8 second = 8 Hz , 1/8 = 0.125 s = 125 ms
 PE_1_2_3 ^= 0x0F; // toggle
 delay(dutyCycle*thePartTwoOfDutyCycle);
 }
 }
}
return 0;
}
```

```
void initialize_PortE(void)
{
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOE;
while(!( SYSCTL_PRGPIO_R& 0x10)){ } //waiting to enable clk

GPIO_PORTE_LOCK_R = 0x4C4F434B; // unlock GPIO Port E
 GPIO_PORTE_CR_R = 0x1F; // allow changes to PE-0
 // only PF0 needs to be unlocked, other bits can't be locked
GPIO_PORTE_DIR_R = 0x0F; //01111 IOOOO
 //GPIO_PORTE_PUR_R = 0x00; // NOT enable pull-up on PF0 and PF4 (RXTERNAL PULL UP)
GPIO_PORTE_DEN_R = 0x1F; //0011111 PE1 , PE2 , PE3 , PE4 ACTIVE

//GPIO_PORTE_DATA_R = 0x00; // reset to initialize all pins

PE_1_2_3=0x00; //specified address
}

//@Def : to read User User_Switch1
//gives value when switch1 is pressed
unsigned long readSwitch1()
{
return (PE4&0x10); // 0 b 0001 0000
}

//Our mcu is working on 80 MHz
// Specified Delay Function
// @Def :
// 1 ms delay function nearly
// if sec equals to 10 this means 1 duty cycle
void delay(int sec)
{
int x=80; //x=(8000);// I delete two zeros for specify process
while(sec>0)
{
 sec--;
 while(x>0)
 {
 x--;
 }
}
}
```

Thanks for reading, in addition I have put my video link on the second page.