Embedded Systems Design

LAB 2 :

1. **Learn how to use GPIO Pins as input.**
2. **Learn how to prevent debouncing and detect level changes (edges) on the pin.**
3. **Polling inputs and control outputs accordingly.**

Group No.: 01

Dixshant Jha	**EE23MT025**
Kaushik Powar	**EE23MT011**

Faculty: **Dr. Abhijit Kshirsagar**

AIM:

1. Read the state on the switches and accordingly control the state of the on-chip LED.
2. Upon each button press change the on-chip LED colour sequentially and also implement debouncing.

MATERIALS REQUIRED:

1. TM4C123GH6PM microcontroller.
2. Code Composer Studio.
3. TM4C123GH6PM Datasheet.

Part 1

Procedure:

1. Configure the corresponding LEDs to Digital Output Mode.
2. Configure the on-chip user switch to Digital Input Mode.
3. Turn on the LED when the user switch is pressed.

Code :

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

int main(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;      /* enable clock to GPIOF */
    GPIO_PORTF_LOCK_R = 0x4C4F434B;    /* unlock commit register */
    GPIO_PORTF_CR_R = 0x1F;            /* make PORTF0 configurable */
    GPIO_PORTF_DEN_R = 0x1E;           /* set PORTF pins 4 pin */
    GPIO_PORTF_DIR_R = 0x0E;           /* set PORTF4 pin as input user switch pin */
    GPIO_PORTF_PUR_R = 0x10;           /* PORTF4 is pulled up */

    while(1)
    {
    if (GPIO_PORTF_DATA_R &= 0x10)     //button isnt pressed
        {
        GPIO_PORTF_DATA_R = 0X00;      /* dark */
        }
    else                    //button is pressed
        {
        GPIO_PORTF_DATA_R = 0X0E;      /* white */
        }
    return 0;
}
```

Results:

Figure 1: LED off (button not pressed)

Figure 2: LED ON (button is pressed)

When the user switch, present on the board, is pressed the RED LED is turned ON, releasing the button turns off the LED. It means the LED is level triggered, by continuously polling the input value on the user switch we determine the state of the LED.

PART 2:

Procedure:

1. We change the LED colour sequentially (R-B-G) on each press of the button, by detecting the level change(edge).
2. Using a delay function after each button press, to prevenet debouncing, we give a 0.5ms delay using a for loop.
3. Also, we use variables prev and present and if prev = present we know that the button was pressed and it has not been released yet. Using this we implement positive edge triggering.

Code:

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

int main(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;      /* enable clock to GPIOF */
    GPIO_PORTF_LOCK_R = 0x4C4F434B;    /* unlock commit register */
    GPIO_PORTF_CR_R = 0x1F;            /* make PORTF0 configurable */
    GPIO_PORTF_DEN_R = 0x1E;           /* set PORTF pins 4 pin */
    GPIO_PORTF_DIR_R = 0x0E;           /* set PORTF4 pin as input user switch pin */
    GPIO_PORTF_PUR_R = 0x10;           /* PORTF4 is pulled up */

    int i;
    int x=0;
    int prev=0,present=0;           /*to detect the edge*/
    while(1){
    present = GPIO_PORTF_DATA_R & 0x10;

    if(prev==0x10 & present==0x0)     /*a leading edge is detected*/
      {
      x=x+1;                  /*sequrntial setting of the LEDs*/
      for(i=0;i<800;i++){}        /* debouncing */
      }
    if(x%3==0)
      {GPIO_PORTF_DATA_R = 0x02;}
    if(x%3==1)
      {GPIO_PORTF_DATA_R = 0x04;}
    if(x%3==2)
      {GPIO_PORTF_DATA_R = 0x08;}
    prev=present;
    }
      return 0;
}
```

Results:

Figure 3: RED LED ON

Figure 4: GREEN LED ON

Figure 5: BLUE LED ON

On pressing the user switch the colour of the LED changes sequentially (Red-Blue-Green). The change in LED colour is implemented on the leading edge of input, on the user switch.

It means the change in LED colour is edge triggered, it is implemented in CCS by using a prev (previous) and present values of the input in the code. A delay of 0.5ms is also given after each edge (leading edge) is detected to implement debouncing, the delay is implemented using the software method (for loop).