EE690
Embedded Systems Design

Group11: EE23DP001 Aditya Shirodkar, EE23MS006 Prasanth Pithanisetty

EK-TM4C123: Toggling On-Board LEDs using On-Board User Switch and incorporation of debounce

Aim: To use individual GPIO pins as inputs as well as outputs

Procedure:

1. Configure GPIO pins corresponding to on-board LEDs as output pins and on-board user switches as input pins
2. Detect positive edge triggering on the user switch
3. Toggle on-board LED whenever user switch is activated
4. Perform steps 1 to 2, and cycle on-board LEDS such that each switch activation triggers a colour change in a cyclic manner
5. Incorporate debounce to eliminate unnecessary toggling of LED

Documents Referred:

1. TM4C123GH6PM microcontroller datasheet
2. Cortex-M4 Technical Reference Manual

Case 1: LED is triggered whenever switch is pressed

The GPIO pin corresponding to the user switch is configured as digital input, whereas the GPIO pin corresponding to the LED is configured as output. As long as the switch is pressed, the LED should be on. It is to be noted that the switch is hardwired such that it is in active low configuration.

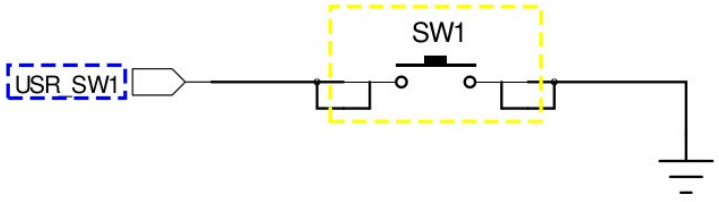

Figure 1: Active Low Configuration of Switch

Code:

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

#define Red_Led 1
#define Blue_Led 2
#define Green_Led 3


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 = 0x1F;             /* set PORTF pins 4-3-2-
1-0 as digital pins */
     GPIO_PORTF_PUR_R = 0x11;             /* enable pull up for pin
4 and 0 */
     GPIO_PORTF_DIR_R = 0x0E;           /* set PORTF3+PORTF2+PORTF1
pin as output (LED) pin and UsrSw1-UsrSw2 as input */


     while(1){
         if(!(GPIO_PORTF_DATA_R & 0x10))           //Check if only
UsrSw1 is set
          {
            GPIO_PORTF_DATA_R |= (1 << Red_Led);        //Only RED
LED on
          }
     else
          {
            GPIO_PORTF_DATA_R &= (0 << Red_Led);        //Only RED
LED Off
          }
     }
}
```

Results:

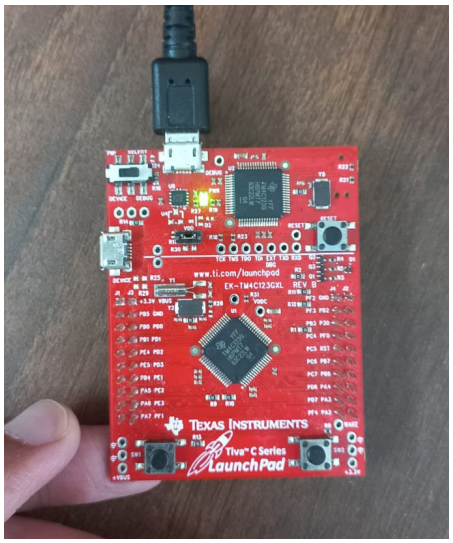

Figure 2: User Switch Off

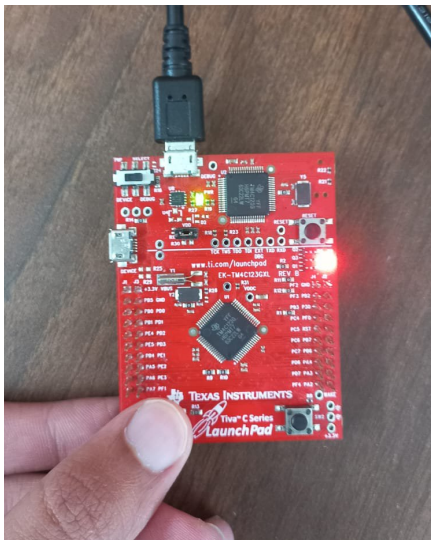

Figure 3: User Switch On

It can be observed that the red LED is triggered only upon activation of the user switch by the user. The LED is to be on whenever the switch is active, making the LED activation a switch-level triggered action. Thus, incorporation of debouncing is not necessary in this case.

Case 2: LEDs are cycled (Red to Blue to Green) upon activation of user switch

Edge Triggering:
The LED is to be toggled only when the user switch is pressed. Upon keeping the switch pressed, or even on releasing the switch, no changes should be reflected in the LED

Debounce:
In order to prevent unnecessary cycling of LEDs due to transients created by activating the user switch, a short delay is implemented, and only the steady state of the switch is read as input. As the switch is to be activated/pressed by the user, a sufficiently large delay is implemented before reading the switch state. The delay duration is such that it is large enough to ignore switch bounce, while being unnoticeable to the user.

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 = 0x1F;            /* set PORTF pins 4-3-2-
1-0 as digital pins */
    GPIO_PORTF_PUR_R = 0x11;            /* enable pull up for pin
4 and 0 */
    GPIO_PORTF_DIR_R = 0x0E;            /* set
PORTF3+PORTF2+PORTF1 pin as output (LED) pin and UsrSw1-UsrSw2 as
input */


    int sw_old=0;
    int sw_new=0;
    int i=1;
    int j;

    while(1){
        for(j = 0; j <5000; j++){}               //Debounce Delay

        sw_new= !(GPIO_PORTF_DATA_R & 0x10);    //Take current
switch value

        if(sw_new>sw_old)                       //Check for
positive edge trigger
        {
            i++;
```

```
            if(i==4)
            {i=1;}
        }

        if(i==1)
        {GPIO_PORTF_DATA_R = 0X02;}             //Red
        if(i==2)
        {GPIO_PORTF_DATA_R = 0X04;}             //Blue
        if(i==3)
        {GPIO_PORTF_DATA_R = 0X08;}             //Green

        sw_old=sw_new;                          //Update old
switch value

    }
}
```

Results:

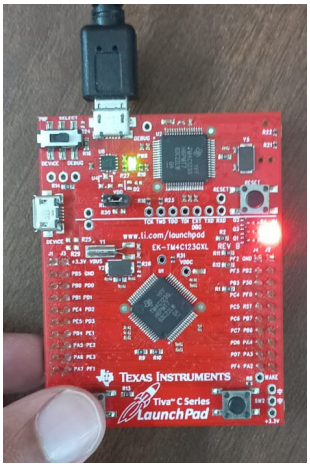

Figure 4: Red LED Glows

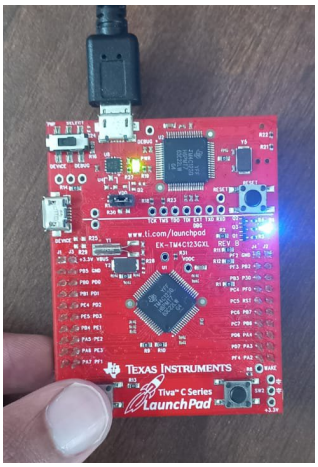

Figure 5: Blue LED Glows

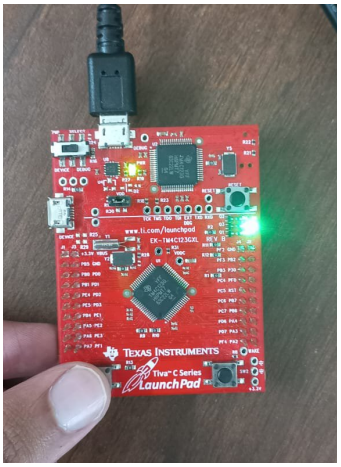

Figure 6: Green LED Glows

It can be observed that as the user switch is activated, the LEDs cycle through the colours as Red, Blue and Green, as depicted in Fig. 4-6 respectively. It is to be noted that the LEDs are programmed to cycle at the positive edge trigger of the switch. As LED action occurs on edge triggering of user switch, debouncing becomes necessary to mitigate false triggering of LED colour cycling.

Conclusion:

As the user switches as well as RGB LEDs are present on the same register (PortF), reading of individual data bit for the user switch using "& 0x10" or "& 0x01", depending upon the user switch being employed, becomes an integral part of the algorithm. Positive edge triggering allows for more control upon LED actions when compared to level triggering, as it allows for the LED to remain on even upon releasing the switch. Incorporation of debounce is a necessity to prevent unnecessary cycling of the LEDs upon activation of user switch.