Assignment 4 Report

EE615 : Embedded Systems Lab

Atharv Gade (210020004)
Yash Meshram (210020053)
5 September 2024

Aim:

Write a program where the main loop monitors the state of a GPIO input pin (SW1) and turns on RED LED whenever SW1 is pressed (and turns off RED LED whenever SW1 is released). The interrupt handler should control the blue LED and toggle it every 500ms i.e. create a 1Hz blink rate of the blue LED.

Theory:

In this experiment, we were supposed to write a code where whenever SW1 is pressed, the RED LED should turn on and turn off when released. In addition to this, we had to program the interrupt handler such that it toggles the BLUE LED every 500ms to create a blink rate of 1Hz.

To implement this we used the systick timer in its interrupt mode. The SysTick timer in the Tiva C microcontroller is a 24-bit down counter that provides an easy way to generate time delays. We will configure it to generate an interrupt every 500 ms, which will be used to toggle the blue LED to create a 1 Hz blinking rate.

Simultaneously, we will monitor the state of the SW1 switch (connected to GPIO pin PF4) to control the red LED. If the switch is pressed, the red LED will turn on; otherwise, the blue LED will continue blinking at a 1 Hz rate.

Code Explanation:

1. SysTick Timer:

- The SysTick timer is set up to generate an interrupt every 500 ms. We use a reload value for the 16 MHz clock such that the timer overflows after 500 ms.

- The blue LED will toggle its state (ON/OFF) each time the interrupt occurs, resulting in a 1 Hz blinking rate.

2. GPIO Input (SW1):

- The SW1 button is connected to GPIO Port F, pin 4 (PF4). It is monitored in the main loop.

- If SW1 is pressed (logic 0), the red LED is turned ON. Otherwise, the red LED is OFF, and the blue LED toggles its state based on the timer interrupt.

3. Startup CCS file:

- The code for the SystickHandler is changed inside the tm4c123gh6pm_startup_ccs.c file is changed to implement the code. To run the code correctly, the startup file is included in the zip file for submission.

Code:

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

/* SysTick memory-mapped registers */
#define STCTRL *((volatile long *) 0xE000E010) // control and status
#define STRELOAD *((volatile long *) 0xE000E014) // reload value
#define STCURRENT *((volatile long *) 0xE000E018) // current value

#define COUNT_FLAG (1 << 16) // bit 16 of CSR automatically set to 1
// when timer expires
#define ENABLE (1 << 0) // bit 0 of CSR to enable the timer
#define CLKINT (1 << 2) // bit 2 of CSR to specify CPU clock
#define TICKINT (1 << 1) //bit 1 of CSR to enable interrupt

#define CLOCK_MHZ 16

void SysTickHandler(void){
    return;
}

void Delay(int ms)
{

STRELOAD = ms*CLOCK_MHZ*10*10*10; // reload value for 'ms' milliseconds


STCTRL |= (CLKINT | ENABLE | TICKINT ); // set internal clock, enable
the timer

while ((STCTRL & COUNT_FLAG) == 0) // wait until flag is set
{
; // do nothing
}

STCTRL = 0; // stop the timer
SysTickHandler();

return;
}



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) == 0)
            {
                GPIO_PORTF_DATA_R |= 0x02;  // Turn on Red LED (PF1)
            }
            else
            {
                GPIO_PORTF_DATA_R = 0x04;
                if ((GPIO_PORTF_DATA_R & 0x10) == 0)
                            {
                                GPIO_PORTF_DATA_R |= 0x02;  // Turn on
Red LED (PF1)
                            }
                else{
                Delay (500);}
                GPIO_PORTF_DATA_R = 0x00;
                if ((GPIO_PORTF_DATA_R & 0x10) == 0)
                                            {
                                                GPIO_PORTF_DATA_R |=
0x02;  // Turn on Red LED (PF1)
                                            }
                                else{
                Delay (500);}
            }


}
}
```

Result:

In this experiment, we successfully implemented a 1 Hz blinking rate for the blue LED using the system timer (SysTick) in interrupt mode. The red LED responds to the state of the SW1 button. When the button is pressed, the red LED turns on, overriding the blue LED. When the button is released, the blue LED resumes blinking.

Conclusion:

We successfully controlled the blue LED's blink rate using the system timer's interrupt mode and monitored the state of the SW1 button to control the red LED. The correct configuration of the SysTick timer allowed for

accurate time delays, and GPIO input handling was achieved using simple polling.