Assignment 3 Report

EE615 : Embedded Systems Lab

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

Aim:

To generate a waveform with frequency - 1KHz and a duty cycle of 20% using the system timer(Systick) in polled mode.

Theory:

In this experiment, we were supposed to generate a waveform with frequency - 1KHz and a duty cycle of 20% using the system timer(Systick) in polled mode.

In the Tiva microcontroller, we have multiple frequencies at which we can operate the board. For this experiment, we use the PIOSC which

- Multiple clock sources for microcontroller system clock. The following clock sources are provided to the TM4C123GH6PM microcontroller:
  - Precision Internal Oscillator (PIOSC) providing a 16-MHz frequency
    - 16 MHz ±3% across temperature and voltage
    - Can be recalibrated with 7-bit trim resolution to achieve better accuracy (16 MHz ±1%)
    - Software power down control for low power modes
  - Main Oscillator (MOSC): A frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins.
  - Low Frequency Internal Oscillator (LFIOSC): On-chip resource used during power-saving modes
  - Hibernate RTC oscillator (RTCOSC) clock that can be configured to be the 32.768-kHz external oscillator source from the Hibernation (HIB) module or the HIB Low Frequency clock source (HIB LFIOSC), which is located within the Hibernation Module.

provides a frequency of 16MHz as mentioned in the datasheet.

The system timer is essentially a down-counter which counts down from a “reload” value which specifies the delay of the timer. The time taken to count down for the timer is the delay which it provides. When it counts down to 0, the **COUNT_FLAG** is set. Now, we can reload the value again into the **STRELOAD** register.

To achieve a frequency of 1kHz, i.e, a delay of 1ms, we set the value of the **STRELOAD** register such that the delay is 1ms. We define **STRELOAD** register to be equal to ‘ms*CLOCK_MHZ*10*10’. Here, ‘ms’ is the delay we want to generate in milliseconds, ‘CLOCK_MHZ’ is the operating frequency of

our clock(16). The delay is then calculated by dividing the **STRELOAD** value by 16x10^6 (16MHz).

Delay = (STRELOAD/16,000,000)

= ((ms * CLOCK_MHz * 100) / 16,000,000)

1kHz = 1ms

= 0.2ms (ON) + 0.8ms (OFF) —> 20% duty cycle

To generate a duty cycle of 20%, we keep the LED on for 20% of the cycle, i.e., for 0.2ms. We load the GPIO_PORTF_DATA register with 0x04 to turn the blue LED on. As we have defined the input to delay to be of type **int**, we put the ms value to be 2 (as **int** can only take integer values, we multiply the product of ms and CLOCK_MHz by 100). Therefore, the delay generated is :

Delay(2) = (STRELOAD/16,000,000)

= ((2 * 16 * 100) / 16,000,000)

= 0.2 ms

For the rest of the cycle, i.e, for 80% of the cycle, we keep the LED OFF for 0.8ms. Now, we load the GPIO_PORTF_DATA register with the value 0x00 to turn off the LED for the specified delay. We generate the delay of 0.8ms the same way we did above while the LED is ON.

We achieve polling by continuously checking the COUNT_FLAG until it is set.

Code:

```
#include "tm4c123gh6pm.h"
#include <stdint.h>
#include <stdbool.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 CLOCK_MHZ 16

void Delay(int ms)
{
    STRELOAD = ms*CLOCK_MHZ*10*10; // reload value for 'ms' milliseconds
    STCTRL |= (CLKINT | ENABLE); // set internal clock, enable the timer

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

        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)
        {
            GPIO_PORTF_DATA_R = 0x04;
            Delay (2);
            GPIO_PORTF_DATA_R = 0x00;
            Delay (8);
        }
    }
```

Result:

The project helped us understand the working of the system timer of the Tiva microcontroller. The notion of polling the flag register and calculating the STRELOAD value was also understood.

Conclusion:

We were successful in obtaining the correct value for the STRELOAD register to generate a delay of 1ms, i.e, a waveform with a frequency of 1kHz with the use of polling. We also were successful to operate at a duty cycle of 20% by setting the input value of our delay function to be 2 and turning the LED on and setting the input value to be 8 and turning the LED off.