॥ सा विद्या या विमुक्तये ॥
भारतीय प्रौद्योगिकी संस्थान धारवाड
Indian Institute of Technology Dharwad

EE690: Embedded Systems Lab

Lab Assignment 3

Group 11
EE23DP001 Aditya Shirodkar
EE23MS005 Prasanth Pithanisetty

EK-TM4C123: Implementation of System Timer (Systick)

1 Aim:

To implement Systick counter for generating pulses of desired frequency and duty cycle.

2 Procedure:

1. Enable clock to the necessary GPIO port and make the port configurable
2. Set the GPIO pins corresponding to the onboard LEDs as digital output pins
3. Define a delay function and set STCTRL and STRELOAD to their appropriate values
4. Toggle the LED at a desired frequency of 1KHz and duty cycle of 20%.
5. Verify LED toggling using an oscilloscope

3 Documents Referred:

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

4 System Timer:

The ARM Cortex-M4F integrated system timer, SysTick, provides a 24-bit, clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. Systick has 3 registers, all of which can be accessed from the privileged mode.

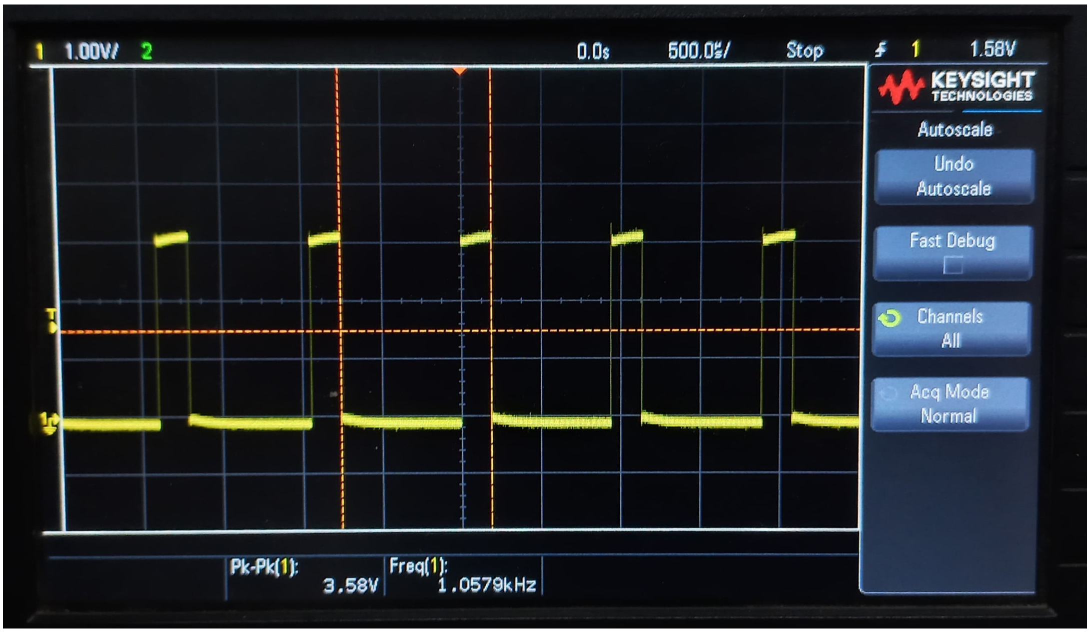
1. SysTick Control and Status Register (STCTRL); which enables Systick features (tm4c123gh6pm datasheet, pg 138)
2. SysTick Reload Value Register (STRELOAD); which specifies the start value to load into the SysTick Current Value (STCURRENT) register when the counter reaches 0 (tm4c123gh6pm datasheet, pg 140).
3. SysTick Current Value Register (STCURRENT); which contains the current value of the SysTick counter (tm4c123gh6pm datasheet, pg 141).

5 Code:

```
#include <stdint.h>
#include <stdbool.h>
#include ”tm4c123gh6pm.h”
#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 us)
{
    STCURRENT = 0;
    STRELOAD = us*16;                // reload value for microseconds
    STCTRL |= (CLKINT | ENABLE);     // set internal clock, enable 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 = 0x1F;            // set PORTF pins 4-3-2-1-0
                                        // as digital pins
    GPIO_PORTF_DIR_R = 0x0E;            // set PORTF3+PORTF2+PORTF1
                                        //pin as output (LED) pin


    while(1){
     GPIO_PORTF_DATA_R = 0X0E;          /* White */
     Delay(200);
     GPIO_PORTF_DATA_R = 0X00;          /* Dark */
     Delay(800);
    }
    return 0;
}
```

6 Results:

As can be seen in Fig. 1, a 1KHz signal is generated with 20% duty ratio is generated on pins 1,2 and 3 of portF, which is verified on an oscilloscope.



Pk-Pk(1): 3.58V

Freq(1): 1.0579kHz

Figure 1: Verification of generated pulses using oscilloscope

7 Conclusion

Use Systick proves to be a highly beneficial method for generating delays/counts, as compared to a simple while() or for() loop, as it frees the processor to perform other activities while the Systick independently counts down to produce the desired value of delay required by the user.