Lab 3: Using SysTick Timer in TIVA μC

Objective: This experiment deals with creating specific time delay and time interval measurement using SysTick timer. You will blink LED with specified time delay and measure the time interval between switching on and off.

Components: Tiva Launchpad, LEDs, switch, 470 ohm resistor, wires.

Note: Please use the sample project file provided along with this document.

SysTick Timer:

- SysTick is a simple counter that we can use to create time delays and generate periodic interrupts.
- 24-bit down counter that runs at the bus clock frequency.
- Tiva can operate up to 80Mhz, the PLL can be used for setting different clock frequencies we require.

The timer consists of three registers:

- SysTick Control and Status (STCTRL): A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- SysTick Reload Value (STRELOAD): The reload value for the counter, used to provide the counter's wrap value.
- SysTick Current Value (STCURRENT): The current value of the counter.

Address	31-24	23-17	16	15-3	2	1	0	Name
SE000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
SE000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
SE000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

Working:

A 80 MHz system clock is given to the timer. Hence, each count takes a time of

$$T = \frac{1}{f} = \frac{1}{80M} = 12.5\, nsec$$

$$Reload\ value = \frac{Desired\ time\ delay}{12.5\, nsec}$$

Register Description:

- **ST_CTRL**

This bit is set to 1 when one cycle of count is over

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															COUNT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													CLK_SRC	INTEN	ENABLE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

This bit is set to 1 to enable system clock

This bit is set to 1 to enable auto interrupt

This bit is set to 1 to enable counter

- **ST_RELOAD**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								RELOAD							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RELOAD															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **ST_CURRENT**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								CURRENT							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CURRENT															
Type	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **NVIC_SYS_PRI3_R**

000 to 111 indicate priority, lowest number highest priority

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TICK			reserved					PENDSV			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEBUG			reserved				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Initializing SysTick Timer:

- Step1: Clear the ENABLE bit to turn off SysTick during initialization.
- Step2: Set the RELOAD register(NVIC_ST_RELOAD_R= Reload value).
- Step3: Clear the counter reg current(NVIC_ST_CURRENT_R = 0).
- Step4: Set desired mode of control register.

  NVIC_ST_CTRL_R = 0x00000005;   counter mode
  NVIC_ST_CTRL_R = 0x00000007;   interrupt mode

Counter mode doesn't interfere with the other execution of the program while Interrupt does. While **Interrupt mode** does run in parallel with main.

```
void SysTick_Init(void)
{
        NVIC_ST_CTRL_R = 0; // 1) disable SysTick during setup
        NVIC_ST_RELOAD_R = 0x00FFFFFF; // 2) maximum reload value
        NVIC_ST_CURRENT_R = 0; // 3) any write to current clears it
        NVIC_ST_CTRL_R = 0x00000005; // 4) enable SysTick with core clock
}
```

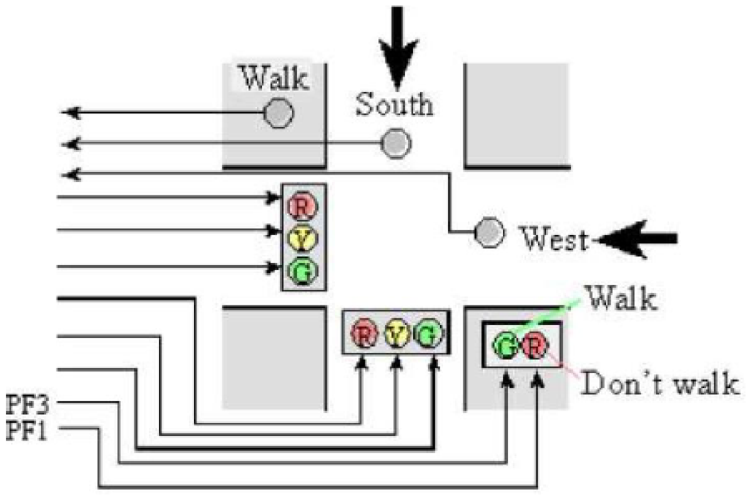
Exercise 3-1: Using SysTick timer, generate a square wave of 500us and observe in debugger. You can connect GPIO pin to ear phone to listen tuning fork sound.

(i) Using counter method
(ii) Using interrupt method

Exercise 3-2: Using SysTick timer, provide a delay of 0.2s to blink the LED.

(i) Using counter method
(ii) Using interrupt method

Exercise 3-3: Design a traffic light controller for the following situation



For West Road:
- Red signal should last for 5s
- Yellow signal should last for 0.2s.
- Green signal duration should be 6.7s
- West road walk for 5s, don't walk for 6.9s

For South Road:
- Red signal should last for 5s
- Yellow signal should last for 0.2s.
- Green signal duration should be 5.2s
- South road walk for 6.5s, don't walk for 5.4s

Reference form: Embedded Systems Shape the World – Valvano : Page 126, Section 9.2