2. Measuring the Time elapsed of single gpio button press.

It is important to choose a good debouncing time for filtering the noise, usually 5 to 30 ms works well for filtering purpose, we will choose 5ms debouncing time.

The implementation used for debouncing is to use a timer and interrupt to test the state of the switch. The idea is to start a timer inside one of the falling or rising edge handler based on the state variable. On every 10^{th} invocation of timer handler, we will calculate the start or stop time based on certain flag conditions. For start/stop time, we will use the current cycle tick provided by CYCCNT register. CYCCNT register is a register in ARM Cortex M4 which provides the tick elapsed since reset. So once we calculate the start and stop time, time elapsed can be measured as difference between stop and start time and subtracting the debouncing time.

The Timer0 with prescaler of 32, counter TOP value of 250 and CLK running at 16MHz, gives an periodic timer interval after every 0.5ms.

```
/****************** Code Snippet ***********************/

        Bool state = true;          / * State variable to ensure that we don't start timer inside external
                                    interrupt handler */

        Bool rising_edge = false;   /* This will be set by rising edge interrupt handler */

        Bool falling_edge = false;  /* This will be set by falling edge interrupt handler */

        Uint8_t timer_count = 10;          //This variable will be 0 after 5 ms


//Get cycle count will fetch the clock ticks since the microcontroller was booted from CYCCNT register

#define get_cycle_count()   *((volatile uint32_t*)0xE0001004)              // Get value from CYCCNT register


Uint32_t start_time = 0;

Bool start_flag = false;

Uint32_t stop_time = 0;

Bool stop_flag = false;

Uint32_t time_elapsed = 0;


//Handler function which is called when Timer 0 of periodic time interval 0.5 expires

Void Timer0_OV_Handler()

{
```

```
If(timer_count > 0)
{
	timer_count--;
}

else
{
	//Reload timer_count with value 10
	timer_count = 10;
	//Make the state variable as true, 5 ms elapsed and stop the timer
	state = true;
	TIM_Base_Stop_IT(&htim1);



//If falling_edge variable is true and GPIO is low -> Valid input after debouncing
If( (falling_edge == true) && GPIO_ReadPin(Push_Button_GPIO_Port, Push_Button_Pin) == 0)
{
	If(!start_flag)
	{
		start_time = get_cycle_count();
		start_flag = true;
		stop_flag = false;
	}
	Else if(!stop_flag)
	{
		Stop_time = get_cycle_count();
		Stop_flag = true;
		Start_flag = true;
		time_elapsed = ((start_time – stop_time)/(16000000)) – 0.0005;
	}
```

```
}
//If falling_edge variable is true and GPIO is low -> Valid input after debouncing
Else If( (rising_edge == true) && GPIO_ReadPin(Push_Button_GPIO_Port,
Push_Button_Pin) == 1)
{
      If(!start_flag)
      {
            start_time = get_cycle_count();
            start_flag = true;
            stop_flag = false;
      }
      Else if(!stop_flag)
      {
            Stop_time = get_cycle_count();
            Stop_flag = true;
            Start_flag = false;
            time_elapsed = ((start_time – stop_time)/(16000000)) – 0.005;   //Time in sec
      }

}
Falling_edge = false;
Rising_edge = false;
}

}
```

```
//Handler function called during Rising Edge Transition of button
Void pinRising_Handler()
{
      if ( state == true)
```

```
	{
		//Start the Timer
		TIM_Base_Start_IT(&htim1);
		state = false;
		rising_edge = true;
	}
	Else
	{
		//No operation (will consume only one CPU cycle)
		__NOP();
	}
}

//Handler function called during the Falling Edge Transition of button
Void pinFalling_Handler()
{
	if ( state == true)
	{
		//Start the Timer
		TIM_Base_Start_IT(&htim1);
		state = false;
		falling_edge = true;
	}
	Else
	{
		//No operation (will consume only one CPU cycle)
		__NOP();
	}

}
```