Embedded Systems Assignment 2

M22CS007
A.Jyothi

```
#include "stm32f4xx.h" // Including device header file.
void delay(); //declaring the function for 2 seconds delay.

int main()
{

RCC->APB1ENR |= 100; // Enabling Timer 4. It is connected to APB1 bus.
RCC->APB1ENR |= 4;   // setting the clock for APB1 where TIM4 is located

RCC->AHB1ENR |= 110; //Enabling the clock signals to the GPIOB and GPIOA
peripherals. Using B for output, C for input. GPIO ports are connected to AHB1
buses.


GPIOB->MODER |=0x10000000;    //Configuring the pin 14 of GPIOB port as output pin.
Moder is 32 bits, out of which 28th bit will be for pin14. So, enabling it to 1.

GPIOB->ODR^=0x4000;     //set that pin as high, so that led blinks when powered on.
for this, enable pin 14 of output data register (16 bits).01000..0


//done with configuring Timer4 and output port to as gpioB.



TIM4->ARR=64000;  // setting ARR to 64000.

TIM4->PSC=500;    // if ARR=64000, and delay is 2 seconds, I calculated PSC, and It
turned out to be 500. This will reduce clock frequency from 16MHZ to 32KHZ.


//done with ARR,PSC assignments.


TIM4->CNT=0;   // clearing the timer 4 counter
TIM4->CR1=1;   // start timer 4
```

```c
int count=1;

while(1)
{
 if(GPIOC->IDR & 0X2000) // using GPIOC as input data register. push button is
connected to the 13th pin. IDR is 16bit register, so 0x2000 is 13th pin. checking
is push button is on or not.
 {
      if((GPIOC->IDR & 0X2000)!= 0X00)// wait if button pressed.
      {

            GPIOB->ODR ^= 0X0000;
            while(count<=5) // loop till 5 times
            {

              GPIOB->ODR ^=0x4000; // blink led, setting the output pin as high.
              delay(); //2 seconds delay.
              GPIOB->ODR ^=0x4000;// again blink led.
              delay(); //2 seconds delay.
              count=count+1;        // increment the count.
            }
  }

}
}

 return 1;
}

void delay()
{
      TIM4->SR=0;// setting status register to zero, so that no ISR is served in
the meantime.
      TIM4->CR1 |=1; // starting the timer 4.

      while((TIM4->SR & TIM_SR_UIF)==0); //Wait until ARR value becomes zero.
}
```

PSC (pre scalar) calculation:

ARR= (delay) x (frequency of clock pre scalar counter)

ARR is 16bit register. So It can have values from 2^0 to 2^16. Therefore, we can choose any value between it. I have chosen 64000.

This will give the frequency of prescalar counter to be 32000. Which means, frequency of clock is reduced from 16MHz to 32KKz.

PSC = (Frequency of clock)/(frequency of clock pre scalar counter)

We know that Frequency of clock is 16MHz. So, we got PSC as 500.