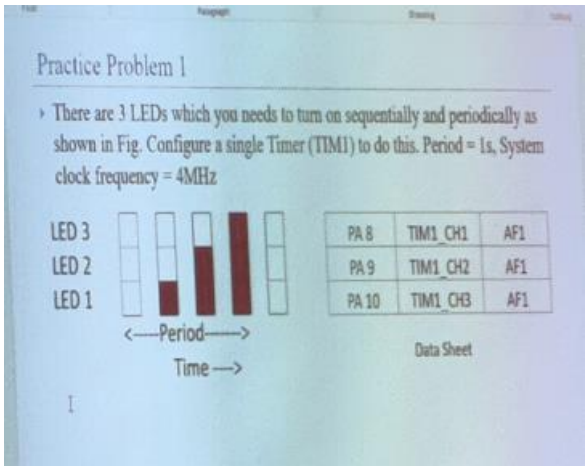
Problem 01:



Code:

```
#include "stm32f446xx.h"


#define SPEAK_PORT GPIOA

#define SPEAK_PIN  0



#define LED_P GPIOA

#define LED_PIN_1   8

#define LED_PIN_2   9

#define LED_PIN_3  10


#define BUTTON_PIN 13


#define VECT_TAB_OFFSET  0x00 /*!< Vector Table base offset field.

                              This value must be a multiple of 0x200. */

/*

 User HSI (high-speed internal) as the processor clock

 See Page 94 on Reference Manual to see the clock tree

 HSI Clock: 16 Mhz, 1% accuracy at 25 oC

 Max Freq of AHB: 84 MHz

 Max Freq of APB2: 84 MHZ

 Max Freq of APB1: 42 MHZ
```

```
 SysTick Clock = AHB Clock / 8

*/


static void LED_PIN_1_Init(){

	RCC->AHB1ENR		|= RCC_AHB1ENR_GPIOAEN;		// Enable GPIOA clock


	// Set mode as Alternative Function 1

		LED_P ->MODER		&= ~(0x03 << (2*LED_PIN_1));			// Clear bits

		LED_P ->MODER		|=   0x02 << (2*LED_PIN_1);		// Input(00), Output(01), AlterFunc(10),
Analog(11)


		LED_P ->AFR[1] &= ~(0xF << (4*LED_PIN_1));	//	AF 1 = TIM2_CH1

		LED_P ->AFR[1] |=   0x1 << (4*LED_PIN_1);	//	AF 1 = TIM2_CH1


		//Set I/O output speed value as very high speed

		LED_P ->OSPEEDR  &= ~(0x03<<(2*LED_PIN_1));				// Speed mask

		LED_P ->OSPEEDR  |=   0x03<<(2*LED_PIN_1);				// Very high speed

		//Set I/O as no pull-up pull-down

		LED_P->PUPDR    &= ~(0x03<<(2*LED_PIN_1));				// No PUPD(00, reset), Pullup(01),
Pulldown(10), Reserved (11)

		//Set I/O as push pull

	//LED_P->OTYPER   &=  ~(1<<LED_PIN_1) ;		// Push-Pull(0, reset), Open-Drain(1)

}

static void LED_PIN_3_Init(){

	RCC->AHB1ENR		|= RCC_AHB1ENR_GPIOAEN;		// Enable GPIOA clock


	// Set mode as Alternative Function 1

		LED_P->MODER		&= ~(0x03 << (2*LED_PIN_3));			// Clear bits

		LED_P->MODER		|=   0x02 << (2*LED_PIN_3);		// Input(00), Output(01), AlterFunc(10),
Analog(11)


		LED_P->AFR[1]  &= ~(0xF << (4*LED_PIN_3));	//	AF 1 = TIM2_CH1

		LED_P->AFR[1]  |=   0x1 << (4*LED_PIN_3);	//	AF 1 = TIM2_CH1
```

```
		//Set I/O output speed value as very high speed
		LED_P->OSPEEDR  &= ~(0x03<<(2*LED_PIN_3));			// Speed mask
		LED_P->OSPEEDR  |=   0x03<<(2*LED_PIN_3);				// Very high speed
		//Set I/O as no pull-up pull-down
		LED_P->PUPDR    &= ~(0x03<<(2*LED_PIN_3));			// No PUPD(00, reset), Pullup(01), Pulldown(10), Reserved (11)
		//Set I/O as push pull
	//LED_P->OTYPER   &=  ~(1<<LED_PIN_1) ;		// Push-Pull(0, reset), Open-Drain(1)
}
static void LED_PIN_2_Init(){
	RCC->AHB1ENR		|= RCC_AHB1ENR_GPIOAEN;		// Enable GPIOA clock

	// Set mode as Alternative Function 1
		LED_P->MODER		&= ~(0x03 << (2*LED_PIN_2));		// Clear bits
		LED_P->MODER		|=   0x02 << (2*LED_PIN_2);		// Input(00), Output(01), AlterFunc(10), Analog(11)

		LED_P->AFR[1]  &= ~(0xF << (4*LED_PIN_2));	//	AF 1 = TIM2_CH1
		LED_P->AFR[1]  |=   0x1 << (4*LED_PIN_2);	//	AF 1 = TIM2_CH1

		//Set I/O output speed value as very high speed
		LED_P->OSPEEDR  &= ~(0x03<<(2*LED_PIN_2));			// Speed mask
		LED_P->OSPEEDR  |=   0x03<<(2*LED_PIN_2);				// Very high speed
		//Set I/O as no pull-up pull-down
		LED_P->PUPDR    &= ~(0x03<<(2*LED_PIN_2));			// No PUPD(00, reset), Pullup(01), Pulldown(10), Reserved (11)
		//Set I/O as push pull
	//LED_P->OTYPER   &=  ~(1<<LED_PIN_1) ;		// Push-Pull(0, reset), Open-Drain(1)
}
static void TIM2_CH1_Init(){
		//tim uptade frequency = TIM_CLK/(TIM_PSC+1)/(TIM_ARR + 1)
	// 4000000 / 40 / 50000 = 2Hz
		// Enable the timer clock
  RCC->APB1ENR		|= RCC_APB1ENR_TIM2EN;		// Enable TIMER clock
```

```
        // Counting direction: 0 = up-counting, 1 = down-counting
        TIM1->CR1 &= ~TIM_CR1_DIR;

TIM1->PSC = 999;      // Prescaler = 23
TIM1->ARR = 3999;  // Auto-reload: Upcouting (0..ARR), Downcouting (ARR..0) 1999
        TIM1->CCMR1 &= ~TIM_CCMR1_OC1M;  // Clear ouput compare mode bits for channel 1
TIM1->CCMR1 |= (TIM_CCMR1_OC1M_0 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2); // OC1M = 0111
TIM1->CCMR1 |= TIM_CCMR1_OC1PE;          // Output 1 preload enable

        TIM1->CCMR1 &= ~TIM_CCMR1_OC2M;  // Clear ouput compare mode bits for channel 1
TIM1->CCMR1 |= (TIM_CCMR1_OC2M_0 | TIM_CCMR1_OC2M_1 | TIM_CCMR1_OC2M_2); // OC1M = 0111
TIM1->CCMR1 |= TIM_CCMR1_OC2PE;          // Output 1 preload enable

        TIM1->CCMR2 &= ~TIM_CCMR2_OC3M;  // Clear ouput compare mode bits for channel 1
TIM1->CCMR2 |= (TIM_CCMR2_OC3M_0 | TIM_CCMR2_OC3M_1 | TIM_CCMR2_OC3M_2); // OC1M = 0011
TIM1->CCMR2 |= TIM_CCMR2_OC3PE;          // Output 1 preload enable

        // Select output polarity: 0 = active high, 1 = active low
        TIM1->CCER |= TIM_CCER_CC1NP; // select active high

// Enable output for ch1
        TIM1->CCER |= TIM_CCER_CC1E;
TIM1->CCER |= TIM_CCER_CC2E;
        TIM1->CCER |= TIM_CCER_CC3E;

// Main output enable (MOE): 0 = Disable, 1 = Enable
        TIM1->BDTR |= TIM_BDTR_MOE;

        TIM1->CCR1  = 1000;     //500
        TIM1->CCR2  = 2000;          //1000
        TIM1->CCR3  = 3000;          //1500
```

```
                TIM1->CR1  |= TIM_CR1_CEN; // Enable counter
}

int main(void){

// Default system clock 4 MHz

        TIM2_CH1_Init();
        LED_PIN_1_Init();
 LED_PIN_2_Init();
 LED_PIN_3_Init();

        while(1);
}
```

LAB Work:

```
#include "stm32f446xx.h"

#define SPEAK_PORT GPIOA
#define SPEAK_PIN  0

int count = 0;
#define LED_P GPIOA
#define LED_PIN    5

#define BUTTON_PIN 13

#define EXTI_PIN 13

#define VECT_TAB_OFFSET  0x00 /*!< Vector Table base offset field.
                This value must be a multiple of 0x200. */
```

```
#define VECT_TAB_OFFSET  0x00 /*!< Vector Table base offset field.
                              This value must be a multiple of 0x200. */

static void LED_Pin_Init(){
        RCC->AHB1ENR                    |= RCC_AHB1ENR_GPIOAEN;          // Enable GPIOA clock

        // Set mode as Alternative Function 1
                LED_P->MODER            &= ~(0x03 << (2*LED_PIN));                  // Clear bits
                LED_P->MODER            |=   0x02 << (2*LED_PIN);                   // Input(00), Output(01), AlterFunc(10),
Analog(11)

                LED_P->AFR[0]  &= ~(0xF << (4*LED_PIN));        //          AF 1 = TIM2_CH1
                LED_P->AFR[0]  |=   0x1 << (4*LED_PIN);         //          AF 1 = TIM2_CH1

                //Set I/O output speed value as very high speed
                LED_P->OSPEEDR  &= ~(0x03<<(2*LED_PIN));                    // Speed mask
                LED_P->OSPEEDR  |=   0x03<<(2*LED_PIN);                             // Very high speed
                //Set I/O as no pull-up pull-down
                LED_P->PUPDR    &= ~(0x03<<(2*LED_PIN));                    // No PUPD(00, reset), Pullup(01),
Pulldown(10), Reserved (11)
                //Set I/O as push pull
        //LED_P->OTYPER   &=  ~(1<<LED_PIN) ;           // Push-Pull(0, reset), Open-Drain(1)
}


static void TIM2_CH1_Init(){
                //tim uptade frequency = TIM_CLK/(TIM_PSC+1)/(TIM_ARR + 1)
        // 4000000 / 40 / 1000 = 100Hz
                // Enable the timer clock
  RCC->APB1ENR                  |= RCC_APB1ENR_TIM2EN;              // Enable TIMER clock

                // Counting direction: 0 = up-counting, 1 = down-counting
```

```
		TIM2->CR1 &= ~TIM_CR1_DIR;

	TIM2->PSC = 39;		// Prescaler = 23
	TIM2->ARR = 1000-1;	// Auto-reload: Upcouting (0..ARR), Downcouting (ARR..0)
		TIM2->CCMR1 &= ~TIM_CCMR1_OC1M;  // Clear ouput compare mode bits for channel 1
	TIM2->CCMR1 |= TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2; // OC1M = 110 for PWM Mode 1 output on ch1
	TIM2->CCMR1 |= TIM_CCMR1_OC1PE;		// Output 1 preload enable

		// Select output polarity: 0 = active high, 1 = active low
		TIM2->CCMR1 |= TIM_CCER_CC1NP; // select active high

	// Enable output for ch1
		TIM2->CCER |= TIM_CCER_CC1E;

	// Main output enable (MOE): 0 = Disable, 1 = Enable
		TIM2->BDTR |= TIM_BDTR_MOE;

		TIM2->CCR1  = 500;		// Output Compare Register for channel 1
		TIM2->CR1  |= TIM_CR1_CEN; // Enable counter
}

static int brightness = 0;

void config_EXTI(void) {
		// GPIO Configuration
	RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;

	// GPIO Mode: Input(00, reset), Output(01), AlterFunc(10), Analog(11, reset)
	GPIOC->MODER &= ~(3UL<<(2*EXTI_PIN)); //input
	GPIOC->MODER &= ~(3UL<<(2*EXTI_PIN));
	// GPIO PUDD: No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
	GPIOC->PUPDR &= ~(3UL<<(2*EXTI_PIN)); // no pull-up, no pull down
```

```
	// Connect External Line to the GPIO
	RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
	SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13;     // SYSCFG external interrupt configuration registers
	SYSCFG->EXTICR[3] |=  SYSCFG_EXTICR4_EXTI13_PC; // port C

	// Ralling trigger selection register (RTSR)
	EXTI->RTSR |= EXTI_RTSR_TR13;  // 0 = disabled, 1 = enabled

	// Interrupt Mask Register (IMR)
	EXTI->IMR |= EXTI_IMR_IM13;     // 0 = marked, 1 = not masked (i.e., enabled)

	// EXIT Interrupt Enable
	NVIC_EnableIRQ(EXTI15_10_IRQn);
 NVIC_SetPriority(EXTI15_10_IRQn, 0); //HIGHEST PRIORITY
	printf("nice\r\n");
}

void EXTI15_10_IRQHandler(void) {
//	NVIC_ClearPendingIRQ(EXTI15_10_IRQn);
	uint32_t j;
	// PR: Pending register
	if (EXTI->PR & EXTI_PR_PR13) {
		// cleared by writing a 1 to this bit
		EXTI->PR |= EXTI_PR_PR13;
		//toggle_LED();
		count = count +1;
		printf("Hi\r\n");
		for(j=0;j<3000;j++);
	}
		    If (count % 3 ==0)
			TIM->CCR1 = 500;
		else if (count % 3 ==1)
			TIM->CCR1=250
```

```
                else if (count % 3 == 2)
                        TIM->CCR1 = 0;
}

int main(void){

        LED_Pin_Init();
        TIM2_CH1_Init(); // Timer to control LED
        config_EXTI();
        while(1){
                EXTI15_10_IRQHandler();
}
```