Problem 1

Using GPIO a LED to send out SOS in Morse code (· · · – – – · · ·) if the user button is pressed. DOT, DOT, DOT, DASH, DASH, DASH, DOT, DOT, DOT. DOT is on for ¼ second and DASH is on for ½ second, with ¼ second between these light-ons. Write the main program below. Demonstrate working code during lab time

Solution

```
#include "stm32f446xx.h"
/* Board name: NUCLEO-F446RE
 PA.5  <--> Green LED (LD2)
 PC.13 <--> Blue user button (B1)
 */
#define LED 5
#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 enable_HSI(){
        /* Enable Power Control clock */
        /* RCC->APB1ENR |= RCC_APB1LPENR_PWRLPEN; */

        // Regulator voltage scaling output selection: Scale 2
        // PWR->CR |= PWR_CR_VOS_1;

        // Enable High Speed Internal Clock (HSI = 16 MHz)
        RCC->CR |= ((uint32_t)RCC_CR_HSION);
        while ((RCC->CR & RCC_CR_HSIRDY) == 0); // Wait until HSI ready

        // Store calibration value
        PWR->CR |= (uint32_t)(16 << 3);

        // Reset CFGR register
        RCC->CFGR = 0x00000000;

        // Reset HSEON, CSSON and PLLON bits
        RCC->CR &= ~(RCC_CR_HSEON | RCC_CR_CSSON | RCC_CR_PLLON);
        while ((RCC->CR & RCC_CR_PLLRDY) != 0); // Wait until PLL disabled

        // Programming PLLCFGR register
        // RCC->PLLCFGR = 0x24003010; // This is the default value

        // Tip:
        // Recommended to set VOC Input f(PLL clock input) / PLLM to 1-2MHz
        // Set VCO output between 192 and 432 MHz,
        // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
        // f(PLL general clock output) = f(VCO clock) / PLLP
        // f(USB OTG FS, SDIO, RNG clock output) = f(VCO clock) / PLLQ
```

```
	RCC->PLLCFGR = 0;

	RCC->PLLCFGR &= ~(RCC_PLLCFGR_PLLSRC);		// PLLSRC = 0 (HSI 16 Mhz clock selected as clock source)

	RCC->PLLCFGR |= 16 << RCC_PLLCFGR_PLLN_Pos;	// PLLM = 16, VCO input clock = 16 MHz / PLLM = 1 MHz

	RCC->PLLCFGR |= 336 << RCC_PLLCFGR_PLLN_Pos;	// PLLN = 336, VCO output clock = 1 MHz * 336 = 336 MHz

	RCC->PLLCFGR |= 4 << RCC_PLLCFGR_PLLP_Pos;	// PLLP = 4, PLLCLK = 336 Mhz / PLLP = 84 MHz

	RCC->PLLCFGR |= 7 << RCC_PLLCFGR_PLLQ_Pos;	// PLLQ = 7, USB Clock = 336 MHz / PLLQ = 48 MHz


	// Enable Main PLL Clock

	RCC->CR |= RCC_CR_PLLON;

	while ((RCC->CR & RCC_CR_PLLRDY) == 0);  // Wait until PLL ready
	// FLASH configuration block

	// enable instruction cache, enable prefetch, set latency to 2WS (3 CPU cycles)

	FLASH->ACR |= FLASH_ACR_ICEN | FLASH_ACR_PRFTEN | FLASH_ACR_LATENCY_2WS;


	// Configure the HCLK, PCLK1 and PCLK2 clocks dividers

	// AHB clock division factor

	RCC->CFGR &= ~RCC_CFGR_HPRE; // 84 MHz, not divided

	// PPRE1: APB Low speed prescaler (APB1)

	RCC->CFGR &= ~RCC_CFGR_PPRE1;

	RCC->CFGR |= RCC_CFGR_PPRE1_DIV2; // 42 MHz, divided by 2

	// PPRE2: APB high-speed prescaler (APB2)

	RCC->CFGR &= ~RCC_CFGR_PPRE2; // 84 MHz, not divided
```

```
        // Select PLL as system clock source
        // 00: HSI oscillator selected as system clock
        // 01: HSE oscillator selected as system clock
        // 10: PLL selected as system clock
        RCC->CFGR &= ~RCC_CFGR_SW;
        RCC->CFGR |= RCC_CFGR_SW_1;
        // while ((RCC->CFGR & RCC_CFGR_SWS_PLL) != RCC_CFGR_SWS_PLL);

        // Configure the Vector Table location add offset address
//      VECT_TAB_OFFSET  = 0x00UL; // Vector Table base offset field.
                    // This value must be a multiple of 0x200.
        SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; // Vector Table Relocation in Internal FLASH

}

static void configure_LED_pin(){
 // Enable the clock to GPIO Port A
 RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;

        // GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
        GPIOA->MODER &= ~(3UL<<(2*LED));
        GPIOA->MODER |=  1UL<<(2*LED);      // Output(01)

        // GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
        GPIOA->OSPEEDR &= ~(3U<<(2*LED));
        GPIOA->OSPEEDR |=  2U<<(2*LED);  // Fast speed

        // GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
        GPIOA->OTYPER &= ~(1U<<LED);     // Push-pull

        // GPIO Push-Pull: No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
        GPIOA->PUPDR  &= ~(3U<<(2*LED));  // No pull-up, no pull-down
        }
```

```
static void configure_PUSH_pin(){
  // Enable the clock to GPIO Port A
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;
        // GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
        GPIOC->MODER &= ~(3UL<<(2*BUTTON_PIN));  // user button=input type declared

        // GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
        GPIOC->OSPEEDR &= ~(3U<<(2*BUTTON_PIN));
        GPIOC->OSPEEDR |=   2U<<(2*BUTTON_PIN);  // Fast speed

        // GPIO Push-Pull: No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
         GPIOC->PUPDR &= ~(3U<<(2*BUTTON_PIN)); // No pull-up, no pull-down
}
static void turn_on_LED(){
        GPIOA->ODR |= (1 << LED);
}
static void turn_off_LED(){
        GPIOA->ODR &= ~(1 << LED);
}
int main(void){
        uint32_t i;
        enable_HSI();
        configure_LED_pin();
        configure_PUSH_pin();
```

```
// Dead loop & program hangs here
        while(1){
                if(!(GPIOC->IDR & 1<<BUTTON_PIN)){
                        turn_on_LED(); //dot
                        for(i=0; i<250000; i++);
                        turn_off_LED();

                        for(i=0; i<500000; i++); //delay

                        turn_on_LED(); //dot
                        for(i=0; i<250000; i++);
                        turn_off_LED();

                        for(i=0; i<500000; i++); //delay

                        turn_on_LED(); //dot
                        for(i=0; i<250000; i++);
                        turn_off_LED();

                        for(i=0; i<500000; i++); //delay

                        turn_on_LED(); //dash
                        for(i=0; i<500000; i++);
                        turn_off_LED();
                        for(i=0; i<500000; i++); //delay

                        turn_on_LED(); //dash
                        for(i=0; i<500000; i++);
                        turn_off_LED();
```

```
                for(i=0; i<500000; i++); //delay

                turn_on_LED(); //dash
                for(i=0; i<500000; i++);
                turn_off_LED();

                for(i=0; i<500000; i++); //delay

                turn_on_LED(); //dot
                for(i=0; i<250000; i++);
                turn_off_LED();

                for(i=0; i<500000; i++); //delay

                turn_on_LED(); //dot
                for(i=0; i<250000; i++);
                turn_off_LED();

                for(i=0; i<500000; i++); //delay

                turn_on_LED(); //dot
                for(i=0; i<250000; i++);
                turn_off_LED();
                for(i=0; i<500000; i++); //delay
        }
} }
```

Problem 2

Write a program that would do the following tasks. Demonstrate the code during lab time. Write the code below: • When push button is not pressed, stepper motor will run counter clock-wise in full-step • When push button is pressed, LED starts blinking AND stepper motor runs clock wise

Solution

```
#include "stm32f446xx.h"


/* Board name: NUCLEO-F446RE
 PA.5  <--> Green LED (LD2)
 PC.13 <--> Blue user button (B1)
 */
#define LED_PIN    5
#define A1 6
#define B1 7
#define A2 8
#define B2 9
#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 enable_HSI(){
        /* Enable Power Control clock */
        /* RCC->APB1ENR |= RCC_APB1LPENR_PWRLPEN; */

        // Regulator voltage scaling output selection: Scale 2
        // PWR->CR |= PWR_CR_VOS_1;
        // Enable High Speed Internal Clock (HSI = 16 MHz)
        RCC->CR |= ((uint32_t)RCC_CR_HSION);
        while ((RCC->CR & RCC_CR_HSIRDY) == 0); // Wait until HSI ready

        // Store calibration value
        PWR->CR |= (uint32_t)(16 << 3);

        // Reset CFGR register
        RCC->CFGR = 0x00000000;

        // Reset HSEON, CSSON and PLLON bits
        RCC->CR &= ~(RCC_CR_HSEON | RCC_CR_CSSON | RCC_CR_PLLON);
        while ((RCC->CR & RCC_CR_PLLRDY) != 0); // Wait until PLL disabled

        // Programming PLLCFGR register
        // RCC->PLLCFGR = 0x24003010; // This is the default value

        // Tip:
        // Recommended to set VOC Input f(PLL clock input) / PLLM to 1-2MHz
        // Set VCO output between 192 and 432 MHz,
        // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
        // f(PLL general clock output) = f(VCO clock) / PLLP
        // f(USB OTG FS, SDIO, RNG clock output) = f(VCO clock) / PLLQ
```

```
	RCC->PLLCFGR = 0;

	RCC->PLLCFGR &= ~(RCC_PLLCFGR_PLLSRC);		// PLLSRC = 0 (HSI 16 Mhz clock selected as clock source)

	RCC->PLLCFGR |= 16 << RCC_PLLCFGR_PLLN_Pos;	// PLLM = 16, VCO input clock = 16 MHz / PLLM = 1 MHz

	RCC->PLLCFGR |= 336 << RCC_PLLCFGR_PLLN_Pos;	// PLLN = 336, VCO output clock = 1 MHz * 336 = 336 MHz

	RCC->PLLCFGR |= 4 << RCC_PLLCFGR_PLLP_Pos;	// PLLP = 4, PLLCLK = 336 Mhz / PLLP = 84 MHz
	RCC->PLLCFGR |= 7 << RCC_PLLCFGR_PLLQ_Pos;	// PLLQ = 7, USB Clock = 336 MHz / PLLQ = 48 MHz


	// Enable Main PLL Clock

	RCC->CR |= RCC_CR_PLLON;

	while ((RCC->CR & RCC_CR_PLLRDY) == 0);  // Wait until PLL ready



	// FLASH configuration block

	// enable instruction cache, enable prefetch, set latency to 2WS (3 CPU cycles)

	FLASH->ACR |= FLASH_ACR_ICEN | FLASH_ACR_PRFTEN | FLASH_ACR_LATENCY_2WS;


	// Configure the HCLK, PCLK1 and PCLK2 clocks dividers

	// AHB clock division factor

	RCC->CFGR &= ~RCC_CFGR_HPRE; // 84 MHz, not divided

	// PPRE1: APB Low speed prescaler (APB1)

	RCC->CFGR &= ~RCC_CFGR_PPRE1;

	RCC->CFGR |= RCC_CFGR_PPRE1_DIV2; // 42 MHz, divided by 2

	// PPRE2: APB high-speed prescaler (APB2)

	RCC->CFGR &= ~RCC_CFGR_PPRE2; // 84 MHz, not divided
	// Select PLL as system clock source

	// 00: HSI oscillator selected as system clock

	// 01: HSE oscillator selected as system clock
```

```
    // 10: PLL selected as system clock
    RCC->CFGR &= ~RCC_CFGR_SW;
    RCC->CFGR |= RCC_CFGR_SW_1;
    // while ((RCC->CFGR & RCC_CFGR_SWS_PLL) != RCC_CFGR_SWS_PLL);

    // Configure the Vector Table location add offset address
//  VECT_TAB_OFFSET  = 0x00UL; // Vector Table base offset field.
                // This value must be a multiple of 0x200.
    SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; // Vector Table Relocation in Internal FLASH

}

static void configure_LED_pin(){
// Enable the clock to GPIO Port A
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;  // coz LED pin is at PA5

    // GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
    GPIOA->MODER &= ~(3UL<<(2*LED_PIN));          //at first,always clear the 2 bits assigned for pin 5
    GPIOA->MODER |=  1UL<<(2*LED_PIN);      //now,set those bits to 01 for Output

    // GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
    GPIOA->OSPEEDR &= ~(3U<<(2*LED_PIN));  //at first,always clear the 2 bits assigned for pin 5
    GPIOA->OSPEEDR |=  2U<<(2*LED_PIN);  // Fast speed

    // GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
    GPIOA->OTYPER &= ~(1U<<LED_PIN);     // Push-pull

    // GPIO Push-Pull: No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
    GPIOA->PUPDR  &= ~(3U<<(2*LED_PIN));  // No pull-up, no pull-down (00) set
    }
```

```
static void turn_on_LED(){
        GPIOA->ODR |= 1U << LED_PIN; // ODR of pin=5 of port-A is set to 1 to turn on LED
}
static void turn_off_LED(){
        GPIOA->ODR &= ~(1U << LED_PIN); //ODR of pin=5 of port-A is set to 0 to turn off LED
}

static void configure_PUSH_pin(){
  // Enable the clock to GPIO Port A
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;
        // GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
        GPIOC->MODER &= ~(3UL<<(2*BUTTON_PIN));  // user button=input type declared

        // GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
        GPIOC->OSPEEDR &= ~(3U<<(2*BUTTON_PIN));
        GPIOC->OSPEEDR |=   2U<<(2*BUTTON_PIN);  // Fast speed

}
static void configure_STEPPER_pin(){
  // Enable the clock to GPIO Port A
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;

        // GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
        GPIOA->MODER &= ~(3UL<<(2*A1));
        GPIOA->MODER |=   1UL<<(2*A1);      // Output(01)

        GPIOA->MODER &= ~(3UL<<(2*B1));
        GPIOA->MODER |=   1UL<<(2*B1);      // Output(01)

        GPIOA->MODER &= ~(3UL<<(2*A2));
        GPIOA->MODER |=   1UL<<(2*A2);      // Output(01)
```

```
    GPIOA->MODER &= ~(3UL<<(2*B2));
    GPIOA->MODER |=  1UL<<(2*B2);      // Output(01)

    // GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
    GPIOA->OSPEEDR &= ~(3U<<(2*A1));
    GPIOA->OSPEEDR |=  2U<<(2*A1);  // Fast speed

    GPIOA->OSPEEDR &= ~(3U<<(2*B1));
    GPIOA->OSPEEDR |=  2U<<(2*B1);  // Fast speed

    GPIOA->OSPEEDR &= ~(3U<<(2*A2));
    GPIOA->OSPEEDR |=  2U<<(2*A2);  // Fast speed

    GPIOA->OSPEEDR &= ~(3U<<(2*B2));
    GPIOA->OSPEEDR |=  2U<<(2*B2);  // Fast speed

    // GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
    GPIOA->OTYPER &= ~(1U<<A1);     // Push-pull
    GPIOA->OTYPER  &=  ~(1U<<B1);    //  Push-pull
    GPIOA->OTYPER  &=  ~(1U<<A2);    //  Push-pull
    GPIOA->OTYPER &= ~(1U<<B2);     // Push-pull

    // GPIO Push-Pull: No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
    GPIOA->PUPDR  &= ~(3U<<(2*A1));  // No pull-up, no pull-down
    GPIOA->PUPDR  &= ~(3U<<(2*B1));  // No pull-up, no pull-down
    GPIOA->PUPDR  &= ~(3U<<(2*A2));  // No pull-up, no pull-down
    GPIOA->PUPDR  &= ~(3U<<(2*B2));  // No pull-up, no pull-down

}
```

```
static void turn_on_A1(){
	GPIOA->ODR |= 1U << A1;
}
static void turn_on_B1(){
	GPIOA->ODR |= 1U << B1;
}
static void turn_on_A2(){
	GPIOA->ODR |= 1U << A2;
}
static void turn_on_B2(){
	GPIOA->ODR |= 1U << B2;
}
static void turn_off_A1(){
	GPIOA->ODR &= ~(1U << A1);
}
static void turn_off_B1(){
	GPIOA->ODR &= ~(1U << B1);
}
static void turn_off_A2(){
	GPIOA->ODR &= ~(1U << A2);
}
static void turn_off_B2(){
	GPIOA->ODR &= ~(1U << B2);
}

int main(void){
	uint32_t i;
	uint32_t delay;
	delay = 100000;
```

```
	enable_HSI();
	configure_STEPPER_pin();
	configure_PUSH_pin();
	turn_off_A1();
	turn_off_B1();
	turn_off_A2();
	turn_off_B2();
	turn_off_LED();


// Dead loop & program hangs here
	while(1){
		if((GPIOC -> IDR & 1UL<<13) != 1UL<<13){ //if button not pressed
		for(i=0; i<delay; i++); // simple delay
			turn_on_A1();
			turn_off_B1();
			turn_off_A2();
			turn_off_B2();

		for(i=0; i<delay; i++); // simple delay
			turn_off_A1();
			turn_on_B1();
			turn_off_A2();
			turn_off_B2();

		for(i=0; i<delay; i++); // simple delay
			turn_off_A1();
			turn_off_B1();
			turn_on_A2();
			turn_off_B2();
```

```
 for(i=0; i<delay; i++); // simple delay
        turn_off_A1();
        turn_off_B1();
        turn_off_A2();
        turn_on_B2();
    }

//pressing push button will energize the 4 coils in reverse order,hence reversing direction of rot
    if((GPIOC -> IDR & 1UL<<13) == 1UL<<13){
        turn_on_LED();
            for(i=0; i<delay; i++); // simple delay
            turn_off_A1();
            turn_off_B1();
            turn_off_A2();
            turn_on_B2();

            for(i=0; i<delay; i++); // simple delay
            turn_off_A1();
            turn_off_B1();
            turn_on_A2();
            turn_off_B2();

            for(i=0; i<delay; i++); // simple delay
            turn_off_A1();
            turn_on_B1();
            turn_off_A2();
            turn_off_B2();

            for(i=0; i<delay; i++); // simple delay
            turn_on_A1();
            turn_off_B1();
            turn_off_A2();
            turn_off_B2();

 }}}
```