

Class 2:

Task: Blink LED Using Push Button

```
1= /**
2  *****
3  * @file      : main.c
4  * @author    : Sagar More
5  * @brief     : Class 2 - Task
6  * @board     : STM32F446RE
7  *
8  *
9  * -----
10 * |      Button      |      LED      |
11 * -----
12 * |   ON Press   | PD12 | PD13 |
13 * -----
14 * |      PC4      |  ON  |  OFF  |
15 * -----
16 * |      PC5      |  OFF |  ON   |
17 * -----
18 * |      PC6      |  ON  |  ON   |
19 * -----
20 *
21 *****
22 */
23
24= /*
25 Register Memory Mapping
26 -----
27 RCC Base Address: 0x40023800
28 RCC_AHB1ENR Offset: 0x30 → RCC_AHB1ENR: 0x40023830
29
30 GPIO Base Address (LED): 0x40020C00
31 GPIO_MODER Offset: 0x00 → GPIO_MODER: 0x40020C00
32 GPIO_ODR Offset: 0x14 → GPIO_ODR: 0x40020C14
33
34 GPIOC Base Address (Switch): 0x40020800
35 GPIOC_MODER Offset: 0x00 → GPIOC_MODER: 0x40020800
36 GPIOC_IDR Offset: 0x10 → GPIOC_IDR: 0x40020810
37
38
39 */
40
41 #include <stdint.h>
42 #include <stdbool.h>
43
44= /* -----
45 * Memory-Mapped Registers
46 * -----*/
47 #define RCC_AHB1ENR (*(volatile uint32_t *)0x40023830) // Clock enable register
48
49 #define GPIO_MODER (*(volatile uint32_t *)0x40020C00) // GPIO mode register
50 #define GPIO_ODR (*(volatile uint32_t *)0x40020C14) // GPIO output register
51
52 #define GPIOC_MODER (*(volatile uint32_t *)0x40020800) // GPIOC mode register
53 #define GPIOC_IDR (*(volatile uint32_t *)0x40020810) // GPIOC input data register
54
```

```

55 /* -----
56  * Main Function
57  * ----- */
58 int main(void)
59 {
60     /* -----
61      * Enable Clock for GPIOC & GPIOD
62      * -----
63      * RCC_AHB1ENR: Bit 2 -> GPIOCEN, Bit 3 -> GPIODEN
64      * 0x5 = (1 << 2) | (1 << 3)
65      */
66     RCC_AHB1ENR |= (1 << 2) | (1 << 3);
67
68     /* -----
69      * Configure GPIOD PIN12 and PIN13 as Output
70      * -----
71      * MODER[24:25] = 01 (PIN12 Output Mode)
72      * MODER[26:27] = 01 (PIN13 Output Mode)
73      */
74
75     GPIOD_MODER |= (1U << 24); // Set MODER10 = 1
76     GPIOD_MODER &= ~(1U << 25); // Set MODER11 = 0
77
78     GPIOD_MODER |= (1U << 26); // Set MODER10 = 1
79     GPIOD_MODER &= ~(1U << 27); // Set MODER11 = 0
80
81     /* -----
82      * Configure GPIOC PIN4, PIN5 and PIN6 as Input
83      * -----
84      * MODER[8:9] = 00 (Input Mode)
85      * MODER[10:11] = 00 (Input Mode)
86      * MODER[12:13] = 00 (Input Mode)
87      */
88
89     // GPIOC Pin4, pin5 and pin6 to input mode
90     GPIOC_MODER &= ~((3U << 8) | (3U << 10) | (3U << 12));
91
92     // Flag to track LED state
93     volatile bool pc4LedFlag = false;
94     volatile bool pc5LedFlag = false;
95     volatile bool pc6LedFlag = false;
96

```

```

97     while (1)
98     {
99         /* Check for the PC4 Button*/
100         pc4LedFlag = ((GPIOC_IDR & (1U << 4)) == 0);
101
102         /* Check for the PC5 Button*/
103         pc5LedFlag = ((GPIOC_IDR & (1U << 5)) == 0);
104
105         /* Check for the PC6 Button*/
106         pc6LedFlag = ((GPIOC_IDR & (1U << 6)) == 0);
107
108         /* -----
109          * Control LED based on Flag
110          * -----*/
111         if (pc4LedFlag){
112             GPIOD_ODR |= (1U << 12);
113             GPIOD_ODR &= ~(1U << 13);
114         }
115         else if (pc5LedFlag){
116             GPIOD_ODR &= ~(1U << 12);
117             GPIOD_ODR |= (1U << 13);
118         }else if (pc6LedFlag){
119             GPIOD_ODR |= (1U << 12);
120             GPIOD_ODR &= ~(1U << 13);
121         }else{
122             GPIOD_ODR &= ~(1U << 12);
123             GPIOD_ODR &= ~(1U << 13);
124         }
125     }
126 }
127

```