

Class 2:

Class Code: Blink LED Using Push Button

STM32F446RE Board

Code:-

```
1  /**
2  ****
3  * @file      : main.c
4  * @author    : Sagar More
5  * @brief     : Class 2 - Blink LED on STM32F446RE by pressing Push Button
6  *
7  * On-board Push Button:
8  *   - Port: GPIOC
9  *   - Pin: 13
10 *   - Bus: AHB1
11 *
12 * On-board LED:
13 *   - Port: GPIOA
14 *   - Pin: 5
15 *   - Bus: AHB1
16 ****
17 */
18
19 /**
20 Register Memory Mapping
21 -----
22 RCC Base Address: 0x40023800
23
24 RCC_AHB1ENR Offset:    0x30
25 RCC_AHB1ENR:          0x40023830
26
27 GPIOA Base Address (LED): 0x40020000
28
29 GPIOA_MODER Offset:    0x00
30 GPIOA_MODER:          0x40020000
31
32 GPIOA_ODR  Offset:    0x14
33 GPIOA_ODR:          0x40020014
34
35 GPIOC Base Address (Switch): 0x40020800
36
37 GPIOC_MODER Offset:    0x00
38 GPIOC_MODER:          0x40020800
39
40 GPIOC_IDR  Offset:    0x10
41 GPIOC_IDR:          0x40020810
42 */
43
```

```

44 #include <stdint.h>
45 #include <stdbool.h>
46
47 /* -----
48  * Memory-Mapped Registers
49  * -----*/
50 #define RCC_AHB1ENR    (*(volatile uint32_t *)0x40023830) // Clock enable register
51
52 #define GPIOA_MODER    (*(volatile uint32_t *)0x40020000) // GPIOA mode register
53 #define GPIOA_ODR      (*(volatile uint32_t *)0x40020014) // GPIOA output data register
54
55 #define GPIOC_MODER    (*(volatile uint32_t *)0x40020800) // GPIOC mode register
56 #define GPIOC_IDR      (*(volatile uint32_t *)0x40020810) // GPIOC input data register
57
58 /* -----
59  * Function Declarations
60  * -----*/
61 void ledHigh(void);
62 void ledLow(void);
63
64 /* -----
65  * Main Function
66  * -----*/
67 int main(void)
68 {
69     /* -----
70      * Enable Clock for GPIOA & GPIOC
71      * -----
72      * RCC_AHB1ENR:
73      * Bit 0 -> GPIOAEN
74      * Bit 2 -> GPIOCEN
75      * 0x5 = (1 << 0) | (1 << 2)
76      */
77     RCC_AHB1ENR |= (0x5U << 0);
78
79     /* -----
80      * Configure GPIOA PIN5 as Output
81      * -----
82      * MODER[11:10] = 01 (Output Mode)
83      */
84     GPIOA_MODER |= (1U << 10); // Set MODER10 = 1
85     GPIOA_MODER &= ~(1U << 11); // Set MODER11 = 0
86

```

```

87  /* -----
88  * Configure GPIOC PIN13 as Input
89  * -----
90  * MODER[27:26] = 00 (Input Mode)
91  */
92  GPIOC_MODER &= ~(1U << 26); // Clear MODER26
93  GPIOC_MODER &= ~(1U << 27); // Clear MODER27
94
95  // Flag to track LED state
96  volatile bool ledFlag = false;
97
98  /* -----
99  * Super Loop
100  * -----*/
101  while (1)
102  {
103      /* -----
104      * Read Button State (PC13)
105      * -----
106      * Active-low button:
107      *   - 0 = Pressed
108      *   - 1 = Released
109      */
110
111      if ((GPIOC_IDR & (1U << 13)) == 0)
112      {
113          ledFlag = true; // Button pressed
114      }
115      else
116      {
117          ledFlag = false; // Button released
118      }
119
120      /* -----
121      * Control LED based on Flag
122      * -----*/
123      if (ledFlag)
124      {
125          ledHigh(); // Turn LED on
126      }
127      else
128      {
129          ledLow(); // Turn LED off
130      }
131  }
132
133  /* -----
134  * LED Control Functions
135  * -----*/
136
137  /* Turn LED ON (PA5 = 1) */
138  void ledHigh(void)
139  {
140      GPIOA_ODR |= (1U << 5);
141  }
142
143  /* Turn LED OFF (PA5 = 0) */
144  void ledLow(void)
145  {
146      GPIOA_ODR &= ~(1U << 5);
147  }
148

```