Electrical Electronics Engineering

Microprocessor Systems

Experiment: 1

Group: 21

Students: Erkan Tiryakioğlu

Alperen Yıldız

Baran Orman

TASKS

PART 1: (no switch bouncing precaution added)

- Initially LED is off.
- After 3 button press LED will blink twice with 100 ms on-off times.
- After 7 button press LED will blink thrice with 300 ms on-off times.
- This procedure goes indefinitely.

CODE:

```c
#include "stm32f4xx_hal.h"
#include "mbed.h"
#include "main.h"

#define BUTTON_PIN GPIO_PIN_0 // Define the pin number for the button
#define BUTTON_PORT GPIOA // Define the GPIO port for the button
#define LED_PIN GPIO_PIN_1 // Define the pin number for the LED
#define LED_PORT GPIOB // Define the GPIO port for the LED

int main(void)
{
  HAL_Init(); // Initialize HAL

  GPIO_InitTypeDef GPIO_InitStruct; // Declare GPIO_InitTypeDef struct

  __HAL_RCC_GPIOA_CLK_ENABLE(); // Enable the clock for the button GPIO port
  __HAL_RCC_GPIOB_CLK_ENABLE(); // Enable the clock for the LED GPIO port

  GPIO_InitStruct.Pin = BUTTON_PIN; // Set the pin number for the button
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT; // Set the pin as input
  GPIO_InitStruct.Pull = GPIO_PULLUP; // Set the pull-up resistor

  HAL_GPIO_Init(BUTTON_PORT, &GPIO_InitStruct); // Initialize the button GPIO port with the above settings

  GPIO_InitStruct.Pin = LED_PIN; // Set the pin number for the LED
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; // Set the pin as output
  GPIO_InitStruct.Pull = GPIO_NOPULL; // Set no pull-up or pull-down resistor
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW; // Set the GPIO speed

  HAL_GPIO_Init(LED_PORT, &GPIO_InitStruct); // Initialize the LED GPIO port with the above settings

  int button_press_count = 0; // Initialize button press count to 0

  while (1) { // Run forever
    if (!HAL_GPIO_ReadPin(BUTTON_PORT, BUTTON_PIN)) { // If the button is pressed (active low)
      button_press_count++; // Increment the button press count

```

```
    if (!HAL_GPIO_ReadPin(BUTTON_PORT, BUTTON_PIN)) { // If the button is pressed (active low)
      button_press_count++; // Increment the button press count

      if (button_press_count == 3) { // If the button has been pressed 3 times
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET); // Turn on the LED
        HAL_Delay(100); // Wait for 100 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); // Turn off the LED
        HAL_Delay(100); // Wait for 100 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET); // Turn on the LED
        HAL_Delay(100); // Wait for 100 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); // Turn off the LED
        HAL_Delay(100); // Wait for 100 ms
      } else if (button_press_count == 7) { // If the button has been pressed 7 times
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET); // Turn on the LED
        HAL_Delay(300); // Wait for 300 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); // Turn off the LED
        HAL_Delay(300); // Wait for 300 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET); // Turn on the LED
        HAL_Delay(300); // Wait for 300 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); // Turn off the LED
        HAL_Delay(300); // Wait for 300 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET); // Turn on the LED
        HAL_Delay(300); // Wait for 300 ms
        HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); // Turn off the LED
        HAL_Delay(300); // Wait for 300
        } else { // If the button has been pressed less than 3 or 7 times
        // Do nothing
      }

      while (!HAL_GPIO_ReadPin(BUTTON_PORT, BUTTON_PIN)) { // Wait for the button to be released
        // Do nothing
      }
    }
  }
}
```

PART 2: Repeat part1 by adding a delay to avoid switch bouncing.

CODE:

```
#include "stm32f4xx_hal.h"
#include "mbed.h"
#include "main.h"

#define BUTTON_PIN GPIO_PIN_0
#define BUTTON_PORT GPIOA
#define LED_PIN GPIO_PIN_1
#define LED_PORT GPIOB

int main(void)
{
  HAL_Init();
  GPIO_InitTypeDef GPIO_InitStruct;

  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  // Button pin configuration
  GPIO_InitStruct.Pin = BUTTON_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(BUTTON_PORT, &GPIO_InitStruct);

  // LED pin configuration
  GPIO_InitStruct.Pin = LED_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LED_PORT, &GPIO_InitStruct);

  uint8_t counter = 0;

  while (1)
  {
    // Read the button input
    if (HAL_GPIO_ReadPin(BUTTON_PORT, BUTTON_PIN) == GPIO_PIN_RESET)
    {
      HAL_Delay(10); // Add debounce delay
```

```
37        HAL_Delay(10); // Add debounce delay
38
39        // Increment the counter on button press
40        counter++;
41
42        // Blink the LED based on the counter value
43        if (counter == 3)
44        {
45          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);
46          HAL_Delay(100);
47          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET);
48          HAL_Delay(100);
49          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);
50          HAL_Delay(100);
51          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET);
52          HAL_Delay(100);
53          counter = 0;
54        }
55        else if (counter == 7)
56        {
57          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);
58          HAL_Delay(300);
59          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET);
60          HAL_Delay(300);
61          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);
62          HAL_Delay(300);
63          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET);
64          HAL_Delay(300);
65          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);
66          HAL_Delay(300);
67          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET);
68          counter = 0;
69        }
70      }
71    }
72  }
73
```

DESCRIPTION

Let's first look at what Part 1 and Part 2 aim at.

In Part 1, the goal is to use an STM32 microcontroller to have an LED blink in a specific way when a button is pressed. According to the specifications given in the assignment, when the button is pressed for the first three times, the LED will blink twice and blink at a time interval of 100 ms. When the button is pressed seven times, the LED will flash three times and flash in a 300 ms interval. This process will continue indefinitely.

In Part 2, the aim is to prevent an electronic problem called switch bouncing. This problem is that when a button is pressed, the microcontroller perceives the button as if it was pressed more than once due to the vibrations caused by the mechanical structure of the button. In this case, errors may occur while the LED is flashing. To avoid switch bouncing, some delay is added after the button is pressed, giving the button time to absorb any unwanted vibrations.

Now, we can explain Part 1 and Part 2 codes.

Part 1:

First, the stm32f4xx_hal.h library is included. This library contains all the necessary definitions for STM32F4 microcontrollers.

Next, two constants are defined, BUTTON_PIN and BUTTON_PORT. These are responsible for the pin and port the button is connected to.

Likewise, two more constants are defined, LED_PIN and LED_PORT. These are responsible for the pin and port the LED is connected to.

The main() function is initialized and the HAL_Init() function is called. This initializes the hardware resources required to run the microcontroller.

A GPIO_InitTypeDef structure named GPIO_InitStruct is defined. This structure contains the configuration settings for the button and LED pins.

Next, the __HAL_RCC_GPIOA_CLK_ENABLE() and __HAL_RCC_GPIOB_CLK_ENABLE() functions are called. This enables the clock signals required for the GPIO ports to be used.

The GPIO_InitStruct structure is first configured for the button pins. Pin property is set to BUTTON_PIN, Mode property is set to GPIO_MODE_INPUT and Pull property is set to GPIO_PULLUP.

Part 2:

Part 2 involves adding a delay to avoid switch bouncing. Switch bouncing is a phenomenon where the mechanical contacts of a switch bounce against each other rapidly when the switch is pressed or released. This can cause multiple

electrical pulses to be generated, which can lead to incorrect behavior in the program.

To avoid switch bouncing, we need to add a delay after each button press. In this implementation, we use the HAL_Delay function provided by the STM32F4xx HAL library to introduce a delay of 50 milliseconds after each button press. This delay ensures that any bouncing of the switch contacts has settled down before the program continues.