Electrical Electronics Engineering

Microprocessor Systems

Experiment: 3

Group: 21

Students: Erkan Tiryakioğlu - 150720051

Alperen Yıldız - 150720067

Baran Orman - 150721063

TASKS

A)

• Use a push button as interrupt input and two LEDs as outputs.

• Use a single ticker function. Ticker counts 1-2-3-4-5-1-2-3-4-5 indefinitely in a period of 1500 ms. Show thiscounting in serial output and toggle LED1 in every 1500 ms.

B)

• Count button press within a 1500ms period. Show press count in serial output.

C)

• If button press counter is greater than 3 within a 1500ms interval, LED2 turns off for 1 sec.

• LED2 is initially on. Turn off period (1 sec) should be determined by a timeout function.

SOLUTION

```
#include "mbed.h"
#include <algorithm>

InterruptIn button(BUTTON1);  // Create an InterruptIn object for BUTTON1
DigitalOut led1(LED1);  // Create a DigitalOut object for controlling LED1
DigitalOut led2(ARDUINO_UNO_D5);  // Create a DigitalOut object for controlling LED2

Ticker ticker;  // Create a Ticker object for periodic tasks
Timeout timeout;  // Create a Timeout object for delayed tasks

volatile int buttonPressCount = 0;  // Variable to keep track of the button press count
volatile bool buttonPressFlag = false;  // Flag to indicate a button press event

void toggleLED1() {
    volatile int count = 0;  // Variable to count the number of LED1 toggles
    led1 = !led1;  // Toggle the state of LED1
    count++;
    if (count > 5) {
        count = 1;  // Reset the count after it reaches 5
    }
    printf("Count: %d\n", count);  // Print the count value
}

void callbackled2(){
    led2 = 1;  // Turn on LED2
    HAL_Delay(1000);  // Delay for 1 second
    led2 = 0;  // Turn off LED2
}

void buttonPress() {
    buttonPressCount++;  // Increment the button press count
    buttonPressFlag = true;  // Set the flag to indicate a button press event
}

void resetButtonPressCount() {
    if (buttonPressCount > 3) {
        timeout.attach(callback(&callbackled2), 1000ms);  // Turn on LED2 after 1 second using a Timeout
    }
    printf("Button Press Count: %d\n", buttonPressCount);  // Print the button press count
    buttonPressCount = 0;  // Reset the button press count
}

int main() {
    button.mode(PullUp);  // Enable the internal pull-up resistor for BUTTON1

    ticker.attach(&toggleLED1, 1500ms);  // Attach the toggleLED1 function to the Ticker with a period of 1.5 seconds
    button.fall(&buttonPress);  // Attach the buttonPress function to the falling edge interrupt of BUTTON1

    while (1) {
        if (buttonPressFlag) {
            timeout.attach(callback(&resetButtonPressCount), 1500ms);  // Call resetButtonPressCount after 1.5 seconds using a Timeout
            buttonPressFlag = false;  // Reset the button press flag
        }
    }
}
```

This code is designed to control LEDs and buttons in a microcontroller-based system. Here's an explanation of its functionality and operation:

At the beginning of the code, the required libraries are included, and an InterruptIn object (button) is created for the BUTTON1 button. DigitalOut objects (led1 and led2) are created to control LED1 and LED2, respectively.

Next, a Ticker object (ticker) is created for periodic tasks, and a Timeout object (timeout) is created for delayed tasks.

The variable buttonPressCount is used to keep track of the button press count, and the boolean variable buttonPressFlag is used to indicate a button press event.

The function toggleLED1() toggles the state of LED1, causing it to blink. It also uses a counter to control the number of blinks.

The callbackled2() function turns on LED2 for 1 second and then turns it off.

The buttonPress() function is triggered when BUTTON1 is pressed. It increments the button press count and sets the buttonPressFlag to true.

The resetButtonPressCount() function checks the value of buttonPressCount. If it is greater than 3, it uses a Timeout object to turn on LED2 after a delay of 1 second. It also resets the button press count and flag.

In the main() function, BUTTON1 is configured in Pull-Up mode. The ticker calls the toggleLED1() function every 1.5 seconds. The button object is assigned the buttonPress() function for the falling edge interrupt of BUTTON1.

Inside an infinite loop, the buttonPressFlag is checked. If the flag is set, the resetButtonPressCount() function is called after a delay of 1.5 seconds using a Timeout object, and the button press count is reset.

This process continues indefinitely, responding to button presses to control the LEDs and perform tasks. LED1 provides visual feedback by blinking at a specific interval. Additionally, there is a delayed task that turns on LED2 when a certain button press count is reached. The code facilitates the interactive control of LEDs and buttons in a microcontroller-based system.