LED ring for meeting room information and timer

Hardware Components:

1. Tiva C Series TM4C123GH6PM Microcontroller

2. Main processing unit for the project, responsible for controlling the WS2812B LEDs based on room status, 16-bit WS2812B 5050 RGB LED Ring Circular development board.

The LED ring is controlled via UART commands to indicate the status of the meeting room. Here are some sample commands:

Key Functions

1. **To reserve a room (Orange LED):**
    - The "O" button indicates that the room is occupied (orange LED).
    - Make sure reservations are only made when the room is Green, or available.
2. **The meeting will begin (blue LED):**
    - The "B" button initiates the meeting.
    - To show the passing of time, set a timer (for example, 16 minutes), during which the LEDs will turn off one after the other.
3. **Yellow LEDs for the Meeting End Indicator:**
    - All of the LEDs turn yellow for five minutes at the end of the timer, signifying that the meeting is over.
4. **Green LED Room Availability:**
    - The room is once again usable when all of the LEDs turn green after five minutes of yellow.

Variable Timing:

Receive time input through UART and set a timer for the LED status to change after the given duration.

Time Data Format:

A standard format, like HH:MM, can be used to send the time since the meeting began.

WS2812B Board to Microcontroller Connection:

- **DIN (Data In)** of the WS2812B LEDs should be connected to a GPIO pin on the microcontroller capable of outputting a data signal with the timing required by the WS2812B protocol.
- **VCC** and **GND** should be connected to a 5V power supply and ground, respectively.

Tiva C Microcontroller Pin Connections:

- **Data Line (DIN)**: Connect to a GPIO pin, e.g., **Pin PA2**.
- **Power (VCC)**: Connect to a 5V pin.
- **Ground (GND)**: Connect to a ground (GND) pin.

Microcontroller programming

Code which glows led

```
#include <stdint.h>
#include "tm4c123gh6pm.h" // Make sure this is correctly included
#define LED_PIN 0x08 // PortA, Pin 3
#define SYS_CLOCK 16000000 // Default 16 MHz

// Function prototypes
void delayMs(uint32_t ms);
void sendBit(uint8_t bit);
void sendByte(uint8_t byte);
void sendColor(uint8_t green, uint8_t red, uint8_t blue);
void initGPIO(void);
void initSysTick(void);
void SysTickDelay(uint32_t ticks);

int main(void) {
int i; // Ensure 'i' is properly declared
initGPIO();
initSysTick();

while (1) {
// Glow green, red, and blue sequentially for all LEDs
```

```
        for (i = 0; i < 16; i++) {  // Correctly declared loop
                        variable 'i'
                    sendColor(0xFF, 0x00, 0x00); // Green
                                }
                    delayMs(500); // Delay for 500ms

        for (i = 0; i < 16; i++) {  // Correctly declared loop
                        variable 'i'
                     sendColor(0x00, 0xFF, 0x00); // Red
                                }
                    delayMs(500); // Delay for 500ms

        for (i = 0; i < 16; i++) {  // Correctly declared loop
                        variable 'i'
                    sendColor(0x00, 0x00, 0xFF); // Blue
                                }
                    delayMs(500); // Delay for 500ms
                                }
                            }

                    void initGPIO(void) {
        SYSCTL_RCGCGPIO_R |= 0x01;  // Enable PortA clock
while ((SYSCTL_PRGPIO_R & 0x01) == 0) {} // Wait for GPIOA to be
                                ready
        GPIO_PORTA_DIR_R |= LED_PIN; // Set PA3 as output
GPIO_PORTA_DEN_R |= LED_PIN; // Enable PA3 digital functionality
                                }

                    void initSysTick(void) {
 NVIC_ST_CTRL_R = 0;              // Disable SysTick during setup
   NVIC_ST_RELOAD_R = 0xFFFFFF;     // Set maximum reload value
     NVIC_ST_CURRENT_R = 0;           // Clear current value
  NVIC_ST_CTRL_R = 0x05;           // Enable SysTick with system
                                clock
                                }

            void SysTickDelay(uint32_t ticks) {
       NVIC_ST_RELOAD_R = ticks - 1;   // Set reload value
      NVIC_ST_CURRENT_R = 0;           // Clear current value
while ((NVIC_ST_CTRL_R & 0x10000) == 0) {} // Wait for COUNT flag
                                }

                void delayMs(uint32_t ms) {
              uint32_t ticks = (SYS_CLOCK / 1000) * ms;
                        SysTickDelay(ticks);
```

```
}

void sendBit(uint8_t bit) {
    if (bit) {
        GPIO_PORTA_DATA_R |= LED_PIN;  // HIGH
        SysTickDelay(13);              // Approx. 800ns
        GPIO_PORTA_DATA_R &= ~LED_PIN; // LOW
        SysTickDelay(7);               // Approx. 450ns
    } else {
        GPIO_PORTA_DATA_R |= LED_PIN;  // HIGH
        SysTickDelay(6);               // Approx. 400ns
        GPIO_PORTA_DATA_R &= ~LED_PIN; // LOW
        SysTickDelay(13);              // Approx. 850ns
    }
}

void sendByte(uint8_t byte) {
    int i; // Declare the loop variable 'i' here
    for (i = 7; i >= 0; i--) { // Send 8 bits, MSB first
        sendBit((byte >> i) & 0x01);
    }
}

void sendColor(uint8_t green, uint8_t red, uint8_t blue) {
    sendByte(green); // Send Green byte
    sendByte(red);   // Send Red byte
    sendByte(blue);  // Send Blue byte
}
```

New code:

Here is an example code to control the WS2812B RGB LEDs using the inbuilt UART of the Tiva C TM4C123GH6PM microcontroller. This code is written for Code Composer Studio and assumes you have the appropriate WS2812B protocol implemented using GPIO and UART for data input.

```
#include <stdint.h>
#include <stdbool.h>
#include "tm4c123gh6pm.h"

// UART and SysTick initialization
void UART_Init(void);
void SysTick_Init(void);
void SysTick_DelayUs(uint32_t delay);
void WS2812B_SendColor(uint8_t red, uint8_t green, uint8_t blue);
void WS2812B_SendReset(void);

// WS2812B Parameters
#define NUM_LEDS 16
#define LED_PIN  (1U << 2) // Assuming PF2 is connected to the data
pin of WS2812B

// LED Color Buffer
uint8_t ledBuffer[NUM_LEDS][3]; // Format: [Green, Red, Blue]

// GPIO Initialization for WS2812B
void GPIO_Init(void) {
  SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R5; // Enable clock for
GPIOF
    while ((SYSCTL_PRGPIO_R & SYSCTL_PRGPIO_R5) == 0);
    GPIO_PORTF_DIR_R |= LED_PIN;  // Set PF2 as output
  GPIO_PORTF_DEN_R |= LED_PIN;  // Enable PF2 digital functionality
    GPIO_PORTF_DATA_R &= ~LED_PIN; // Initialize pin to low
}

int main(void) {
    char input;
uint8_t red = 0, green = 0, blue = 0;

    UART_Init();
  SysTick_Init();
    GPIO_Init();

    while (1) {
  // Wait for data from UART
input = UART0_DR_R; // Read received character

    // Example: Change colors based on keyboard input
  if (input == 'R') { red = 255; green = 0; blue = 0; } // Red
    if (input == 'G') { red = 0; green = 255; blue = 0; } //
Green
  if (input == 'B') { red = 0; green = 0; blue = 255; } // Blue

  // Update all LEDs with the new color
  for (int i = 0; i < NUM_LEDS; i++) {
      ledBuffer[i][0] = green;
```

```
                        ledBuffer[i][1] = red;
                        ledBuffer[i][2] = blue;
                                }

                        // Send colors to WS2812B
                    for (int i = 0; i < NUM_LEDS; i++) {
                WS2812B_SendColor(ledBuffer[i][1], ledBuffer[i][0],
                        ledBuffer[i][2]);
                                }
                          WS2812B_SendReset();
                                }
                              }

                    // Function to initialize UART0
                          void UART_Init(void) {
       SYSCTL_RCGCUART_R |= SYSCTL_RCGCUART_R0;  // Enable UART0
     SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R0;  // Enable PORTA clock
        UART0_CTL_R &= ~UART_CTL_UARTEN;          // Disable UART0
    UART0_IBRD_R = 104;                       // Set baud rate to 9600
                           UART0_FBRD_R = 11;
     UART0_LCRH_R = UART_LCRH_WLEN_8;         // 8-bit, no parity, 1
                               stop bit
        UART0_CTL_R |= UART_CTL_UARTEN;           // Enable UART0
      GPIO_PORTA_AFSEL_R |= 0x03;               // Enable PA0, PA1
                          alternate functions
      GPIO_PORTA_DEN_R |= 0x03;                 // Enable PA0, PA1
                           digital functions
    GPIO_PORTA_PCTL_R |= 0x11;                // Set PA0, PA1 for UART
                                }

                    // Function to initialize SysTick
                          void SysTick_Init(void) {
              NVIC_ST_CTRL_R = 0;          // Disable SysTick
             NVIC_ST_RELOAD_R = 0xFFFFFF; // Max reload value
           NVIC_ST_CURRENT_R = 0;       // Clear current value
       NVIC_ST_CTRL_R = 5;          // Enable SysTick with processor
                                 clock
                                }

                    // Microsecond delay using SysTick
                void SysTick_DelayUs(uint32_t delay) {
         NVIC_ST_RELOAD_R = (delay * 16) - 1; // For 16 MHz clock
                           NVIC_ST_CURRENT_R = 0;
     while ((NVIC_ST_CTRL_R & 0x10000) == 0); // Wait for COUNT flag
                                }

               // Function to send color data to WS2812B
void WS2812B_SendColor(uint8_t red, uint8_t green, uint8_t blue) {
                    uint8_t color[3] = {green, red, blue};
                         for (int j = 0; j < 3; j++) {
                           for (int i = 7; i >= 0; i--) {
                               if (color[j] & (1 << i)) {
```

```
    GPIO_PORTF_DATA_R |= LED_PIN;  // High for '1'
       SysTick_DelayUs(0.9);         // ~900 ns
         GPIO_PORTF_DATA_R &= ~LED_PIN; // Low
       SysTick_DelayUs(0.35);        // ~350 ns
                    } else {
    GPIO_PORTF_DATA_R |= LED_PIN;  // High for '0'
       SysTick_DelayUs(0.35);        // ~350 ns
         GPIO_PORTF_DATA_R &= ~LED_PIN; // Low
       SysTick_DelayUs(0.9);         // ~900 ns
                      }
                    }
                  }
                }

   // Function to send reset signal
    void WS2812B_SendReset(void) {
      GPIO_PORTF_DATA_R &= ~LED_PIN;
SysTick_DelayUs(80); // >50us reset pulse
                }
```