Objective: Develop a C program for a microcontroller that simulates an advanced LED control system. The system should use nested structures to represent detailed control aspects of LEDs, including state, brightness, and color, as well as group settings for multiple LEDs.

Code:

```c
#include <stdio.h>
#include <stdint.h>

// Structure Definitions

// LEDSettings structure represents the state, brightness, and color of an individual LED.
typedef struct {
    uint8_t state;      // ON/OFF state (1 for ON, 0 for OFF)
    uint8_t brightness;  // Brightness level (0-255)
    uint32_t color;     // RGB color value (encoded as 0xRRGGBB)
} LEDSettings;

// LEDGroup structure represents a group of LEDs.
// It contains both individual LED settings (singleLED) and group-wide settings (groupState and
groupBrightness).
typedef struct {
    LEDSettings singleLED;    // Individual LED settings
    uint8_t groupState;       // Group state (1 for all ON, 0 for all OFF)
    uint8_t groupBrightness;  // Group brightness adjustment (0-255)
} LEDGroup;

// Function to Initialize LED Group
// This function initializes an LEDGroup structure with default values.
// - All LEDs are turned off (state = 0).
// - Brightness is set to minimum (brightness = 0).
// - No color is assigned (color = 0x000000, which represents black).
void initLEDGroup(LEDGroup *group) {
    // Initialize the individual LED settings
    group->singleLED.state = 0;          // Set LED state to OFF
    group->singleLED.brightness = 0;     // Set LED brightness to minimum
    group->singleLED.color = 0x000000;   // Set LED color to black (no color)

    // Initialize the group settings
    group->groupState = 0;               // Set group state to OFF (all LEDs OFF)
    group->groupBrightness = 0;          // Set group brightness to minimum
}

// Function to Update LED Group Settings
// This function updates both the individual LED settings and the group-wide settings.
// Parameters:
// - group: Pointer to the LEDGroup structure to be updated.
// - groupState: New group state (1 for ON, 0 for OFF).
```

```
// - groupBrightness: New brightness level for the group (0-255).
// - state: New state for the individual LED (1 for ON, 0 for OFF).
// - brightness: New brightness level for the individual LED (0-255).
// - color: New color for the individual LED (RGB encoded as a 32-bit integer).
void updateLEDGroupSettings(LEDGroup *group, uint8_t groupState, uint8_t groupBrightness,
                            uint8_t state, uint8_t brightness, uint32_t color) {
    // Update the individual LED settings
    group->singleLED.state = state;         // Set the new state (ON/OFF) for the individual LED
    group->singleLED.brightness = brightness; // Set the new brightness for the individual LED
    group->singleLED.color = color;         // Set the new RGB color for the individual LED

    // Update the group-wide settings
    group->groupState = groupState;         // Set the new group state (ON/OFF for all LEDs)
    group->groupBrightness = groupBrightness; // Set the new group brightness adjustment
}

// Function to Display LED Group Status
// This function prints the current status of both the individual LED and the group.
// - Displays the state (ON/OFF), brightness, and color of the individual LED.
// - Displays the state (all ON/OFF) and brightness of the LED group.
void displayLEDGroupStatus(const LEDGroup *group) {
    // Display the status of the individual LED
    printf("Individual LED Status:\n");
    printf("State: %s\n", group->singleLED.state ? "ON" : "OFF");  // Check if LED is ON or OFF
    printf("Brightness: %u\n", group->singleLED.brightness);       // Print the LED brightness level (0-255)
    printf("Color (RGB): #%06X\n", group->singleLED.color);        // Print the LED color in RGB format

    // Display the status of the LED group
    printf("\nGroup Status:\n");
    printf("Group State: %s\n", group->groupState ? "All ON" : "All OFF");  // Check if all LEDs in the
group are ON or OFF
    printf("Group Brightness: %u\n", group->groupBrightness);              // Print the group brightness level
(0-255)
}

// Main function for testing the LED control system
int main() {
    LEDGroup ledGroup;  // Create an instance of LEDGroup to store the LED group data

    // Step 1: Initialize the LED group with default values
    initLEDGroup(&ledGroup);

    // Step 2: Display the default LED group status
    printf("Initial LED Group Status:\n");
    displayLEDGroupStatus(&ledGroup);

    // Step 3: Update the LED group settings with new values
```

```
    // Example: Set the group to ON, group brightness to 128, individual LED to ON, brightness to 200, and color to orange (0xFF5733).
    updateLEDGroupSettings(&ledGroup, 1, 128, 1, 200, 0xFF5733);

    // Step 4: Display the updated LED group status
    printf("\nUpdated LED Group Status:\n");
    displayLEDGroupStatus(&ledGroup);

    return 0;  // End of the program
}
```

Explanation:

- LEDSettings Structure: Represents the individual LED's state (ON/OFF), brightness, and color.
- LEDGroup Structure: Contains both individual LED settings and group-wide settings (groupState and groupBrightness).
- initLEDGroup Function: Initializes the LED group to default values (all OFF, brightness = 0, and no color).
- updateLEDGroupSettings Function: Updates both the individual LED and the group's settings based on the function parameters.
- displayLEDGroupStatus Function: Prints the status of both the individual LED and the group, including brightness and color (RGB encoded).
- Main Function: Initializes an LEDGroup, displays the initial settings, updates the settings, and displays the updated status.

Compilation & Testing:

To compile this program in a standard C environment:

1. Save the code in a file named led_control.c.
2. Compile the code using a GCC compiler:

   gcc -o led_control led_control.c

3. Run the compiled program:

   ./led_control

This program will output the initial and updated status of the LED group as per the changes made by the updateLEDGroupSettings function.

For microcontroller environments (e.g., with an embedded IDE like **STM32CubeIDE or Keil**):

- Create a new project for your microcontroller.
- Add this C code to the project files.
- Ensure you have a proper setup for input/output functions (UART or other communication interfaces) for the printf statements to work.
- Build and flash the program to the microcontroller.