```c
/**
* VibrateMotor.c
* This file contains the service to activate/deactivate the button
* vibration motors.
*/

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "VibrateMotor.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"

#define _500_MS 500 // amount of time for which each button should vibrate [ms]

static uint8_t MyPriority; // module-level variable for the VibrateMotor module's priority

/* Prototypes */
bool InitializeVibrateMotor(uint8_t Priority);
ES_Event RunVibrateMotor(ES_Event CurrentEvent);
bool PostVibrateMotor(ES_Event ThisEvent);

/**
* Initializes the VibrateMotor service.
* @param Priority: priority of the VibrateMotor service
*/
bool InitializeVibrateMotor(uint8_t Priority) {
    ES_Event ThisEvent;
    MyPriority = Priority; // set the priority

    ThisEvent.EventType = ES_INIT; // post the initial transition event
    if (ES_PostToService(MyPriority, ThisEvent) == true) {
        return true;
    } else {
        return false;
    }
}

/**
* Runs the VibrateMotor service.
* @param CurrentEvent: the current event that has occurred
*/
ES_Event RunVibrateMotor(ES_Event CurrentEvent) {
    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no problems will occur

    switch (CurrentEvent.EventType) {
        case MotorOn: {
            switch (CurrentEvent.EventParam) {
                case 1: {
                    MOTOR_1_PORT |= MOTOR_1_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
                    //printf("Vibrate button 1\r\n");
                }
                break;
                case 2: {
                    MOTOR_2_PORT |= MOTOR_2_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
```

```
                    //printf("Vibrate button 2\r\n");
                }
                break;
                case 3: {
                    MOTOR_3_PORT |= MOTOR_3_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
                    //printf("Vibrate button 3\r\n");
                }
                break;
                case 4: {
                    MOTOR_4_PORT |= MOTOR_4_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
                    //printf("Vibrate button 4\r\n");
                }
                break;
                case 5: {
                    MOTOR_5_PORT |= MOTOR_5_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
                    //printf("Vibrate button 5\r\n");
                }
                break;
                case 6: {
                    MOTOR_6_PORT |= MOTOR_6_PIN;
                    ES_Timer_InitTimer(VIB_MOTOR_TIMER, _500_MS); // Initialize 500-ms timer
                    //printf("Vibrate button 6\r\n");
                }
                break;
            }
        }
        break;
        case ES_TIMEOUT: {
            if (CurrentEvent.EventParam == VIB_MOTOR_TIMER) {
                /* Ensure that all vibration motors are off */
                MOTOR_1_PORT &= ~MOTOR_1_PIN;
                MOTOR_2_PORT &= ~MOTOR_2_PIN;
                MOTOR_3_PORT &= ~MOTOR_3_PIN;
                MOTOR_4_PORT &= ~MOTOR_4_PIN;
                MOTOR_5_PORT &= ~MOTOR_5_PIN;
                MOTOR_6_PORT &= ~MOTOR_6_PIN;
            }
        }
        break;
    }
    return ReturnEvent;
}

/**
* Posts events to the VibrateMotor service.
* @param ThisEvent: event to post to VibrateMotor servce
* @return true if successfully posted; false otherwise
*/
bool PostVibrateMotor(ES_Event ThisEvent) {
  return ES_PostToService(MyPriority, ThisEvent);
}
```