ใบงานที่ 7

Timer / PWM

อุปกรณ์ที่ต้องใช้ในการทดลอง

1. สาย USB สำหรับโหลดโปรแกรมลงบอร์ด
2. บอร์ด ESP32
3. เครื่องคอมพิวเตอร์
4. Oscilloscope
5. Servo motor (optional)

General Purpose Timer; Alarm Timer

1. Include header ที่เกี่ยวข้อง

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gptimer.h"
#include "esp_log.h"

static const char *TAG = "GPTimer";

typedef struct {
    uint64_t event_count;
} example_queue_element_t;
```

2. โค้ดในส่วนของการ hook ISR ของตัว Timer

```c
static bool IRAM_ATTR example_timer_on_alarm_cb(gptimer_handle_t timer, const gptimer_alarm_event_data_t *edata, void *user_data)
{
    BaseType_t high_task_awoken = pdFALSE;
    QueueHandle_t queue = (QueueHandle_t)user_data; // Retrieve count value and send to queue
    example_queue_element_t ele = {
        .event_count = edata->count_value
    };
    xQueueSendFromISR(queue, &ele, &high_task_awoken);
    // return whether we need to yield at the end of ISR
    return (high_task_awoken == pdTRUE);
}
```

3. ส่วนต้นของการกำหนดค่าพารามิเตอร์ที่ต้องการ configure ให้กับตัว Timer เพื่อทำการนับเวลาจากสัญญาณนาฬิกา

```c
void app_main(void)
{
    example_queue_element_t ele;
    QueueHandle_t queue = xQueueCreate(10, sizeof(example_queue_element_t));
    if (!queue) {
        ESP_LOGE(TAG, "Creating queue failed");
        return;
    }

    ESP_LOGI(TAG, "Create timer handle");
    gptimer_handle_t gptimer = NULL;
    gptimer_config_t timer_config = {
        .clk_src = GPTIMER_CLK_SRC_DEFAULT,
        .direction = GPTIMER_COUNT_UP,
        .resolution_hz = 1000000, // 1MHz, 1 tick=1us
    };
    ESP_ERROR_CHECK(gptimer_new_timer(&timer_config, &gptimer));

    gptimer_event_callbacks_t cbs = {
        .on_alarm = example_timer_on_alarm_cb,
    };

    ESP_ERROR_CHECK(gptimer_register_event_callbacks(gptimer, &cbs, queue));
    ESP_LOGI(TAG, "Enable timer");
    ESP_ERROR_CHECK(gptimer_enable(gptimer));
```

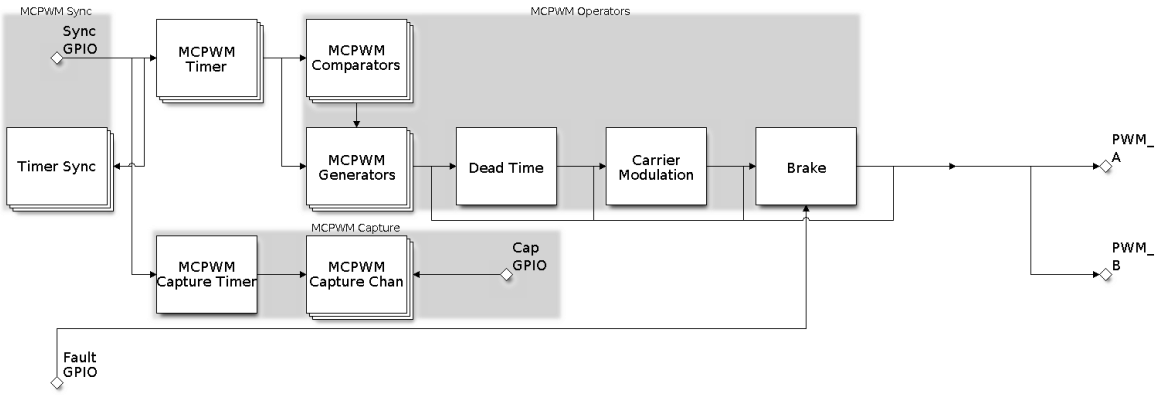
4. โค้ดส่วนหลังเพื่อทำการ configure alarm event ให้กับตัว Timer โดยในที่นี้จะเกิด Interrupt ทุกๆ 1 วินาที และจะทำการโหลดค่าเริ่มต้นใหม่เพื่อวนนับไปจนกว่าจะสั่ง Timer หยุดการทำงาน

```c
    ESP_LOGI(TAG, "Start timer, auto-reload at alarm event");
    gptimer_alarm_config_t alarm_config = {
        .reload_count = 0,
        .alarm_count = 1000000, // period = 1s
        .flags.auto_reload_on_alarm = true,
    };
    ESP_ERROR_CHECK(gptimer_set_alarm_action(gptimer, &alarm_config));
    ESP_ERROR_CHECK(gptimer_start(gptimer));
    int record = 4;
    while (record) {
        if (xQueueReceive(queue, &ele, pdMS_TO_TICKS(2000))) {
            ESP_LOGI(TAG, "Timer reloaded, count=%llu", ele.event_count);
            record--;
        } else {
            ESP_LOGW(TAG, "Missed one count event");
        }
    }
    ESP_LOGI(TAG, "Stop timer");
    ESP_ERROR_CHECK(gptimer_stop(gptimer));

    ESP_LOGI(TAG, "Disable timer");
    ESP_ERROR_CHECK(gptimer_disable(gptimer));
    ESP_LOGI(TAG, "Delete timer");
    ESP_ERROR_CHECK(gptimer_del_timer(gptimer));

    vQueueDelete(queue);
}
```

5. ทำการ monitor ข้อมูลทางหน้าจอคอมพิวเตอร์และ capture หน้าจอแปะผลที่ได้ลงในพื้นที่ว่างด้านล่าง

MCPWM; Motor Control Pulse-Width Modulation

1. Block diagram ของโมดูล MCPWM



1) **MCPWM Timer:** The timer base of the final PWM signal.
2) **MCPWM Operator:** The Key module that is responsible for generating the PWM waveforms. It consists of other submodules, like comparator, PWM generator, dead time, and carrier modulator.
3) **MCPWM Comparator:** The compare module takes the time-base count value as input, and continuously compares it to the threshold value configured. When the timer is equal to any of the threshold values, a compare event will be generated and the MCPWM generator can update its level accordingly.
4) **MCPWM Generator:** One MCPWM generator can generate a pair of PWM waves, complementarily or independently, based on various events triggered by other submodules like MCPWM Timer and MCPWM Comparator.

2. โค้ดในส่วนของการ include header และการประกาศค่าต่างๆ ที่เกี่ยวข้อง รวมทั้งฟังก์ชันในการเปลี่ยนค่า (Map function) จากมุม (angle) เป็นจำนวนนับที่ใช้ในการกำหนดให้กับตัว comparator เปลี่ยนค่า SERVO_PULSE_GPIO เป็น 13 เพื่อกำหนดขาที่ต้องการสร้างสัญญาณเป็น GPIO13

```c
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "driver/mcpwm_prelude.h"

static const char *TAG = "MCPWM";

// Please consult the datasheet of your servo before changing the following parameters
#define SERVO_MIN_PULSEWIDTH_US 500  // Minimum pulse width in microsecond
#define SERVO_MAX_PULSEWIDTH_US 2500  // Maximum pulse width in microsecond
#define SERVO_MIN_DEGREE        -90   // Minimum angle
#define SERVO_MAX_DEGREE        90    // Maximum angle

#define SERVO_PULSE_GPIO            0        // GPIO connects to the PWM signal line
#define SERVO_TIMEBASE_RESOLUTION_HZ 1000000  // 1MHz, 1us per tick
#define SERVO_TIMEBASE_PERIOD        20000    // 20000 ticks, 20ms

static inline uint32_t example_angle_to_compare(int angle)
{
    return (angle - SERVO_MIN_DEGREE) * (SERVO_MAX_PULSEWIDTH_US - SERVO_MIN_PULSEWIDTH_US) / (SERVO_MAX_DEGREE - SERVO_MIN_DEGREE) + SERVO_MIN_PULSEWIDTH_US;
}
```

3. โค้ดส่วนต้นที่ใช้ในการสร้างตัว Time base พร้อมทั้งทำการผูก MCPWM Comparator เข้ากับ MCPWM Operator

```c

void app_main(void)
{
    ESP_LOGI(TAG, "Create timer and operator");
    mcpwm_timer_handle_t timer = NULL;
    mcpwm_timer_config_t timer_config = {
        .group_id = 0,
        .clk_src = MCPWM_TIMER_CLK_SRC_DEFAULT,
        .resolution_hz = SERVO_TIMEBASE_RESOLUTION_HZ, //resolution: 1us
        .period_ticks = SERVO_TIMEBASE_PERIOD, //timer period: 20ms
        .count_mode = MCPWM_TIMER_COUNT_MODE_UP,
    };
    ESP_ERROR_CHECK(mcpwm_new_timer(&timer_config, &timer));

    mcpwm_oper_handle_t oper = NULL;
    mcpwm_operator_config_t operator_config = {
        .group_id = 0, // operator must be in the same group to the timer
    };
    ESP_ERROR_CHECK(mcpwm_new_operator(&operator_config, &oper));

    ESP_LOGI(TAG, "Connect timer and operator");
    ESP_ERROR_CHECK(mcpwm_operator_connect_timer(oper, timer));

    ESP_LOGI(TAG, "Create comparator and generator from the operator");
    mcpwm_cmpr_handle_t comparator = NULL;
    mcpwm_comparator_config_t comparator_config = {
        .flags.update_cmp_on_tez = true,
    };
    ESP_ERROR_CHECK(mcpwm_new_comparator(oper, &comparator_config, &comparator));
```

4. โค้ดส่วนท้ายในการสร้าง MCPWM Generator ส่งสัญญาณออกที่ SERVO_PULSE_GPIO และทำการเชื่อมเข้ากับ MCPWM Operator พร้อมทั้งกำหนด event ในการสร้างสัญญาณดังนี้
   - เมื่อค่าตัวนับของ timer มีค่าเป็น 0 ให้สร้างสัญญาณโลจิก 1
   - เมื่อตัว MCPWM Comparator เปรียบเทียบค่าของ Timer กับค่าที่กำหนดมีค่าเท่ากัน จะทำให้ MCPWM Generator สร้างสัญญาณโลจิก 0

```c
    mcpwm_gen_handle_t generator = NULL;
    mcpwm_generator_config_t generator_config = {
        .gen_gpio_num = SERVO_PULSE_GPIO, //GPIO0
    };
    ESP_ERROR_CHECK(mcpwm_new_generator(oper, &generator_config, &generator));

    // set the initial compare value, so that the servo will spin to the center position
    ESP_ERROR_CHECK(mcpwm_comparator_set_compare_value(comparator, example_angle_to_compare(0)));

    ESP_LOGI(TAG, "Set generator action on timer and compare event");
    // go high on counter empty
    ESP_ERROR_CHECK(mcpwm_generator_set_action_on_timer_event(generator,
                    MCPWM_GEN_TIMER_EVENT_ACTION(MCPWM_TIMER_DIRECTION_UP, MCPWM_TIMER_EVENT_EMPTY, MCPWM_GEN_ACTION_HIGH)));
    // go low on compare threshold
    ESP_ERROR_CHECK(mcpwm_generator_set_action_on_compare_event(generator,
                    MCPWM_GEN_COMPARE_EVENT_ACTION(MCPWM_TIMER_DIRECTION_UP, comparator, MCPWM_GEN_ACTION_LOW)));

    ESP_LOGI(TAG, "Enable and start timer");
    ESP_ERROR_CHECK(mcpwm_timer_enable(timer));
    ESP_ERROR_CHECK(mcpwm_timer_start_stop(timer, MCPWM_TIMER_START_NO_STOP));

    int angle = 0;
    int step = 2;
    while (1) {
        ESP_LOGI(TAG, "Angle of rotation: %d", angle);
        ESP_ERROR_CHECK(mcpwm_comparator_set_compare_value(comparator, example_angle_to_compare(angle)));
        //Add delay, since it takes time for servo to rotate, usually 200ms/60degree rotation under 5V power supply
        vTaskDelay(pdMS_TO_TICKS(500));
        if ((angle + step) > 60 || (angle + step) < -60) {
            step *= -1;
        }
        angle += step;
    }
}
```

5. นำออสซิลโลสโครปจับขาของ GPIO ที่ให้สัญญาณจาก MCPWM จากนั้น save รูปสัญญาณหน้าจอที่ได้ใส่ในพื้นที่ว่างด้านล่าง พร้อมทั้งสังเกตลักษณะของคลื่นที่ได้จาก PWM

ใบงานท้ายบท

ให้นิสิตทำการเขียนโค้ดเพื่อสั่งงานหลอด LED ให้ติดเป็นเวลา 250ms และดับเป็นเวลา 250ms โดยใช้ Timer หรือ MCPWM อย่างใดอย่างหนึ่ง พร้อมทั้ง อธิบายโค้ดในส่วนที่กำหนดค่าเพื่อให้ได้ผลตามที่โจทย์ต้องการและ capture ภาพหน้าจอออสซิลโลสโครปใส่ลงในผลใบงานท้ายบทด้วย