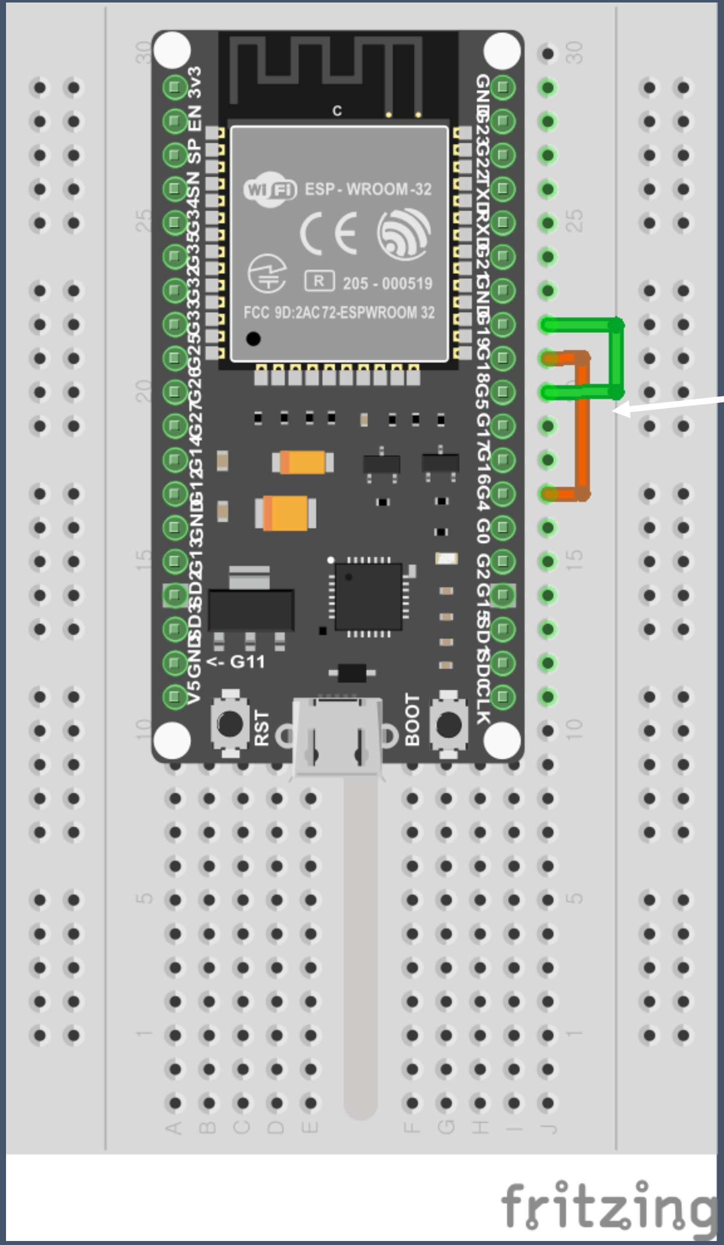
메이커 스페이스
G·캠프
ESP32
온라인 워크숍 초급과정

1. GPIO Input/Output Original



GPIO18 – GPIO4 연결
GPIO19 – GPIO5 연결

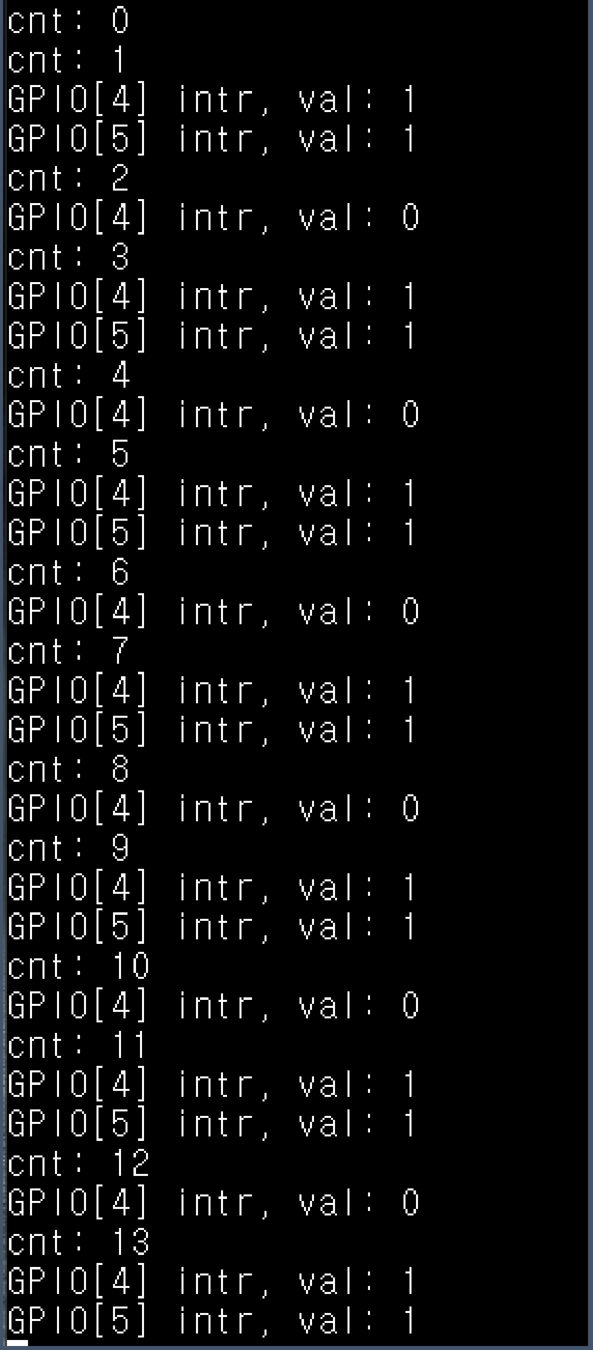
www.CodeZoo.co.kr

1. GPIO Input/Output Original

정확하게 동작 !!

- ✓ GPIO4 ANYEDGE 인터럽트 설정 (Rising, Falling 인터럽트 둘다체크)
- ✓ GPIO5 POSEDGE 인터럽트 설정 (Rising 인터럽트만 체크)

- ✓ GPIO18, GPIO19는 짝수일때는 0 출력 (%2), 홀수일때는 1 출력
- ✓ 인터럽트 설정과 함께 체크해보면 정확하게 신호대로 동작되고 있음



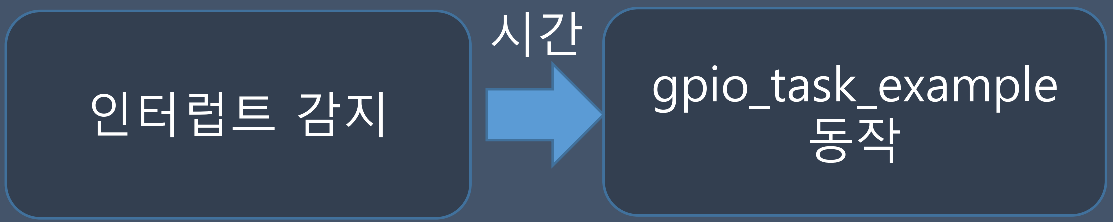
```
cnt: 0
cnt: 1
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 2
GPIO[4] intr, val: 0
cnt: 3
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 4
GPIO[4] intr, val: 0
cnt: 5
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 6
GPIO[4] intr, val: 0
cnt: 7
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 8
GPIO[4] intr, val: 0
cnt: 9
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 10
GPIO[4] intr, val: 0
cnt: 11
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
cnt: 12
GPIO[4] intr, val: 0
cnt: 13
GPIO[4] intr, val: 1
GPIO[5] intr, val: 1
```

2. GPIO 버튼 입력시 핀 상태값이 0/1 이 나오는 이유

인터럽트가 동작한 시간과 상태값을 읽어오는 시간이 다르기 때문

✓ 버튼에서 Falling Interrupt를 체크해서 Event 발생, Event는 EventQueue에 저장
✓ 상태값을 읽어서 출력하는 gpio_task_example 태스크는 gpio_evt_queue를 읽어서
  저장된 이벤트를 가져오고 printf로 출력할때 핀 상태값을 읽음

```
static void gpio_task_example(void* arg)
{
    uint32_t io_num;
    for(;;) {
        if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
            printf("GPIO[%d] intr, val: %d₩n", io_num, gpio_get_level(io_num));
        }
    }
}
```



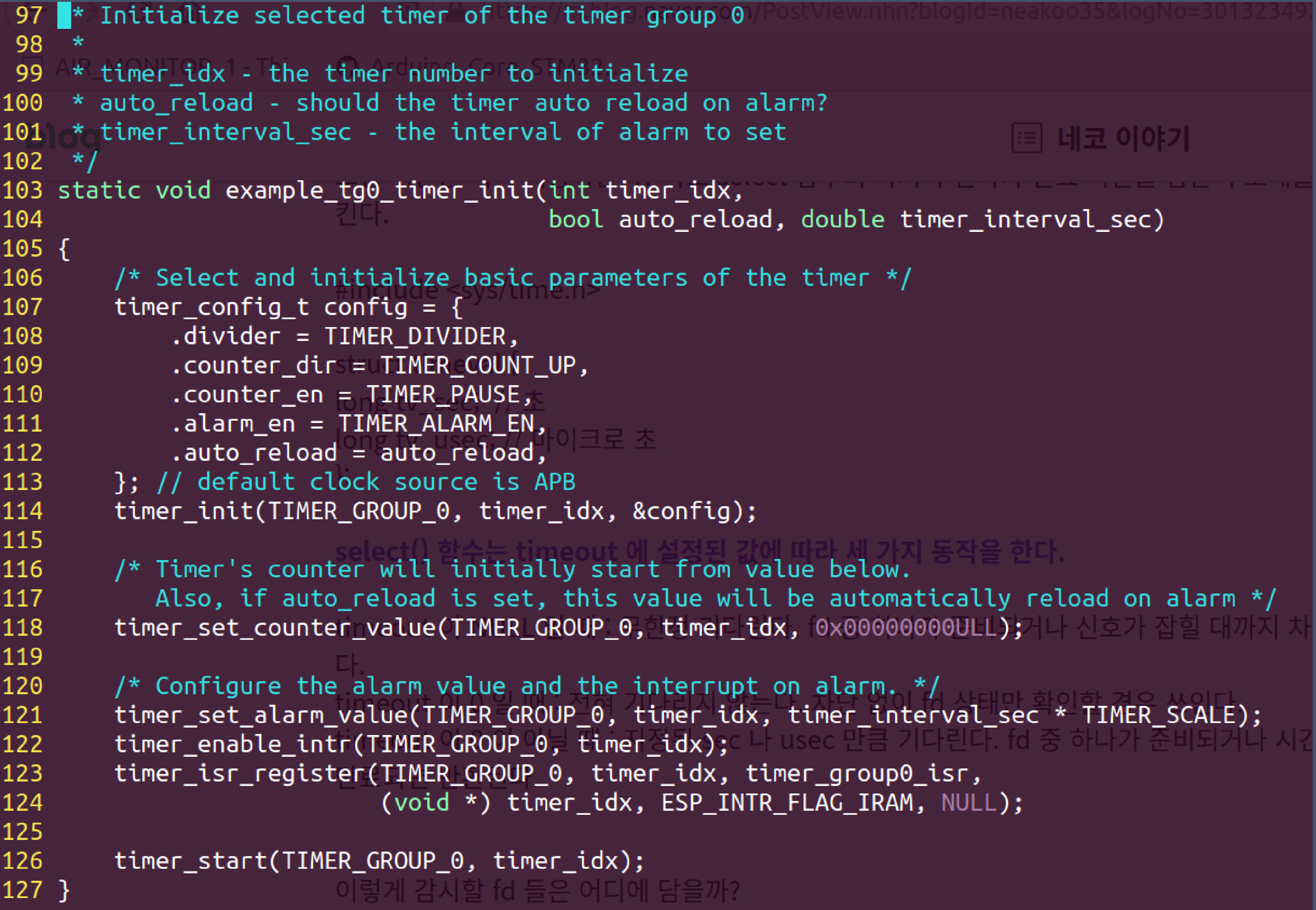
3. ESP32 Timer & Timer Interrupt

- ✓ A 16-bit clock prescaler, from 2 to 65536
- ✓ A 64-bit time-base counter
- ✓ Configurable up/down time-base counter: incrementing or decrementing
- ✓ Halt and resume of time-base counter
- ✓ Auto-reload at alarm
- ✓ Software-controlled instant reload
- ✓ Level and edge interrupt generation

ESP32 64bit Timer 를 다루기 위해서는

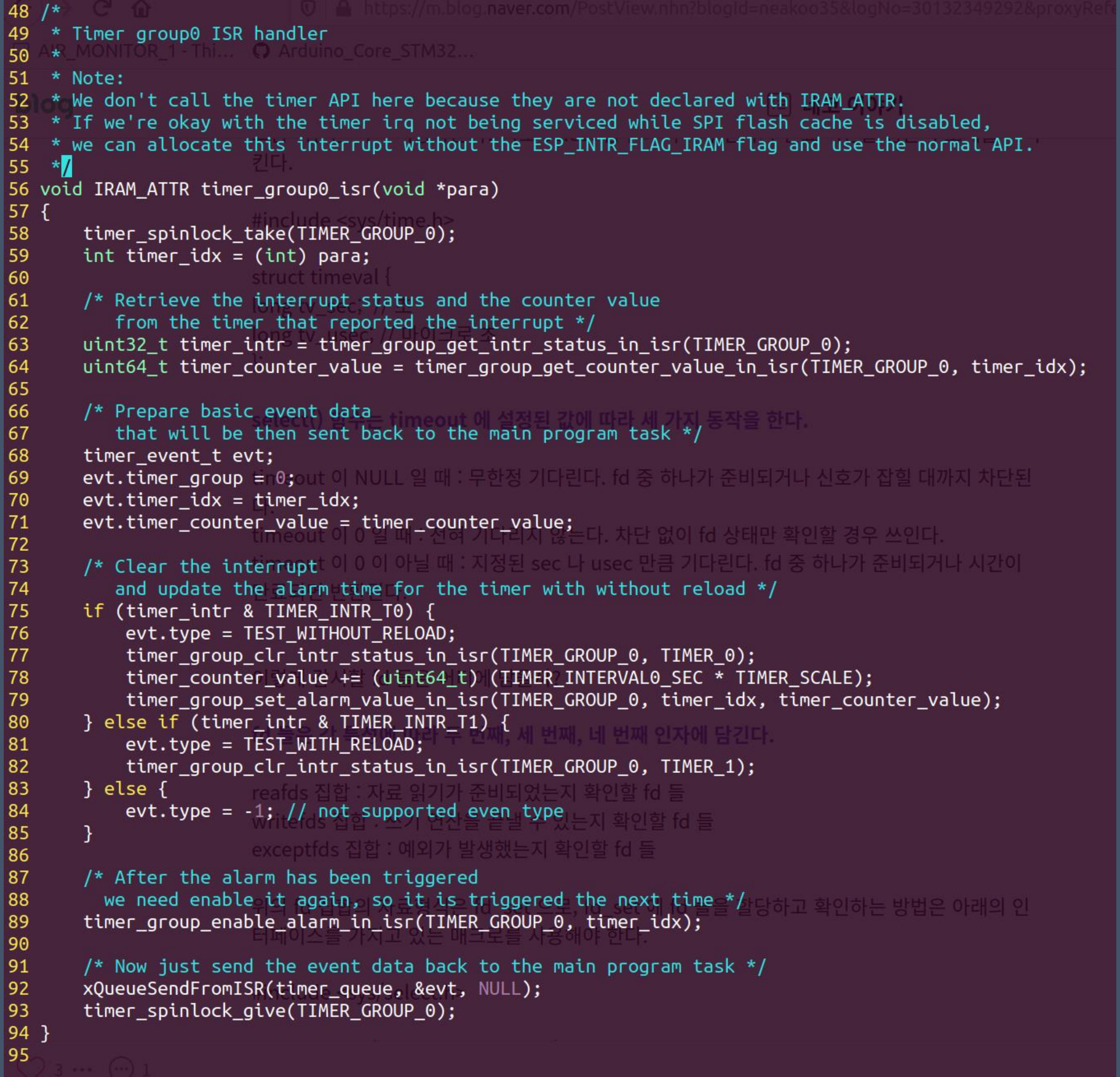
- ✓ 타이머 초기화 -타이머가 작동하도록 설정해야 하는 매개 변수와 타이머 구성에 따라 제공되는 특정 기능
- ✓ 타이머 제어 -타이머 값을 읽고, 타이머를 일시 중지 또는 시작하고, 작동 방식을 변경하는 방법
- ✓ 알람 - 알람 설정 및 사용 방법
- ✓ 인터럽트 - 인터럽트 를 활성화하고 사용하는 방법

✓ 타이머 초기화, 제어



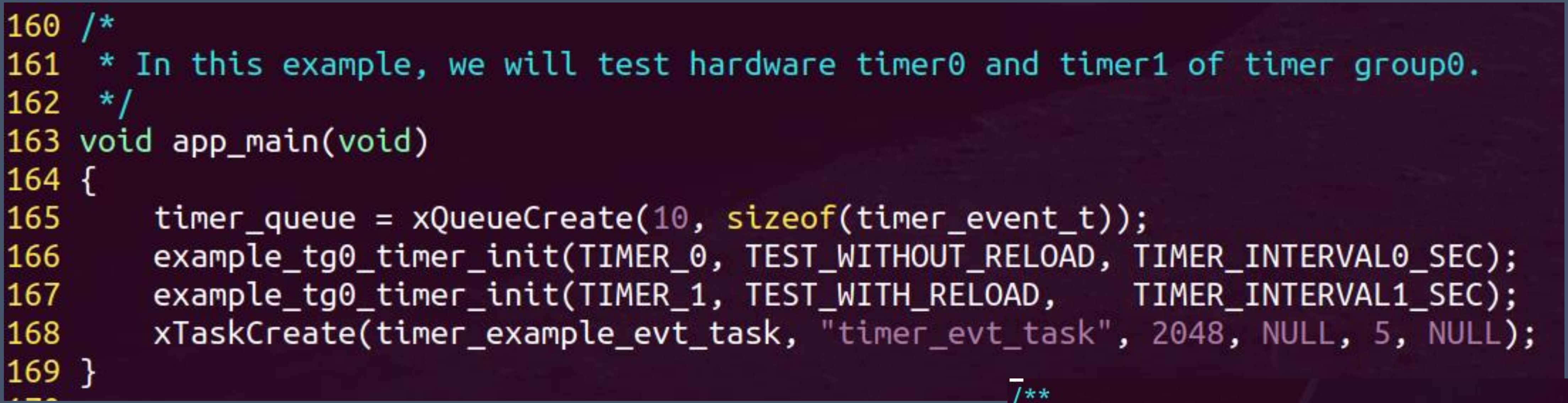
```c
/* Initialize selected timer of the timer group 0
 *
 * timer_idx - the timer number to initialize
 * auto_reload - should the timer auto reload on alarm?
 * timer_interval_sec - the interval of alarm to set
 */
static void example_tg0_timer_init(int timer_idx,
                                   bool auto_reload, double timer_interval_sec)
{
    /* Select and initialize basic parameters of the timer */
    timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_PAUSE,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = auto_reload,
    }; // default clock source is APB
    timer_init(TIMER_GROUP_0, timer_idx, &config);

    /* Timer's counter will initially start from value below.
       Also, if auto_reload is set, this value will be automatically reload on alarm */
    timer_set_counter_value(TIMER_GROUP_0, timer_idx, 0x00000000ULL);

    /* Configure the alarm value and the interrupt on alarm. */
    timer_set_alarm_value(TIMER_GROUP_0, timer_idx, timer_interval_sec * TIMER_SCALE);
    timer_enable_intr(TIMER_GROUP_0, timer_idx);
    timer_isr_register(TIMER_GROUP_0, timer_idx, timer_group0_isr,
                       (void *) timer_idx, ESP_INTR_FLAG_IRAM, NULL);

    timer_start(TIMER_GROUP_0, timer_idx);
}
```

✓ 타이머 알람, 인터럽트

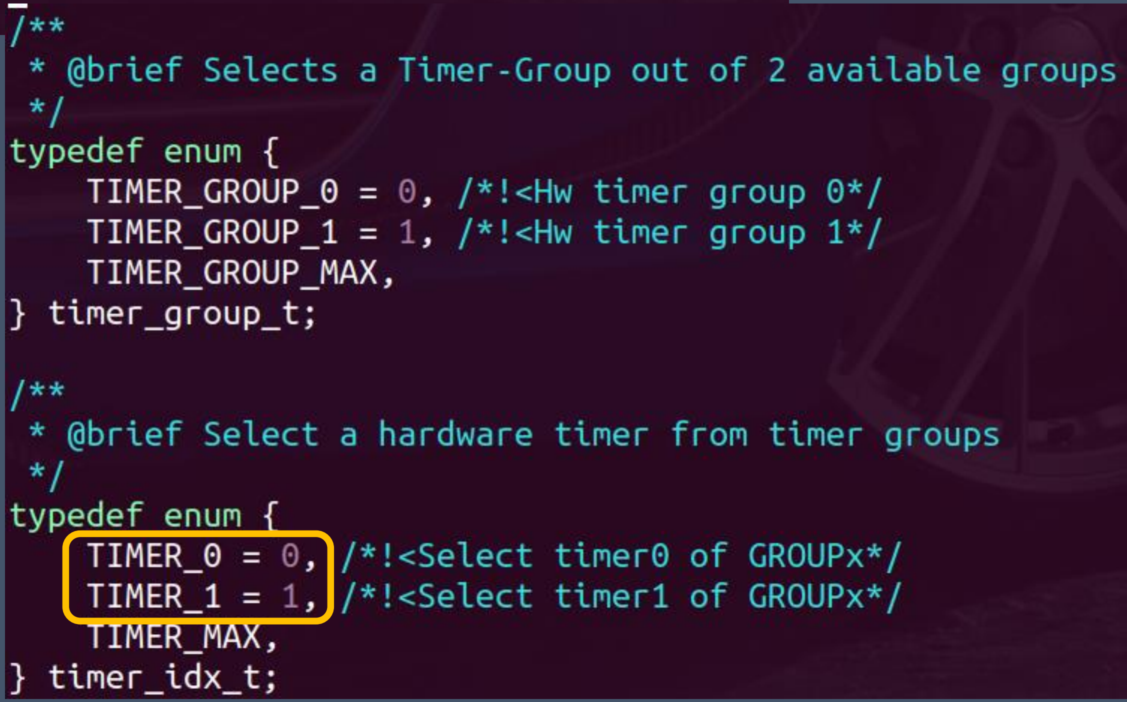


```
/*
 * Timer group0 ISR handler
 *
 * Note:
 * We don't call the timer API here because they are not declared with IRAM_ATTR.
 * If we're okay with the timer irq not being serviced while SPI flash cache is disabled,
 * we can allocate this interrupt without the ESP_INTR_FLAG_IRAM flag and use the normal API.
 */
void IRAM_ATTR timer_group0_isr(void *para)
{
    timer_spinlock_take(TIMER_GROUP_0);
    int timer_idx = (int) para;

    /* Retrieve the interrupt status and the counter value
       from the timer that reported the interrupt */
    uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
    uint64_t timer_counter_value = timer_group_get_counter_value_in_isr(TIMER_GROUP_0, timer_idx);

    /* Prepare basic event data
       that will be then sent back to the main program task */
    timer_event_t evt;
    evt.timer_group = 0;
    evt.timer_idx = timer_idx;
    evt.timer_counter_value = timer_counter_value;

    /* Clear the interrupt
       and update the alarm time for the timer with without reload */
    if (timer_intr & TIMER_INTR_T0) {
        evt.type = TEST_WITHOUT_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
        timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
        timer_group_set_alarm_value_in_isr(TIMER_GROUP_0, timer_idx, timer_counter_value);
    } else if (timer_intr & TIMER_INTR_T1) {
        evt.type = TEST_WITH_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
    } else {
        evt.type = -1; // not supported even type
    }

    /* After the alarm has been triggered
      we need enable it again, so it is triggered the next time */
    timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);

    /* Now just send the event data back to the main program task */
    xQueueSendFromISR(timer_queue, &evt, NULL);
    timer_spinlock_give(TIMER_GROUP_0);
}
```

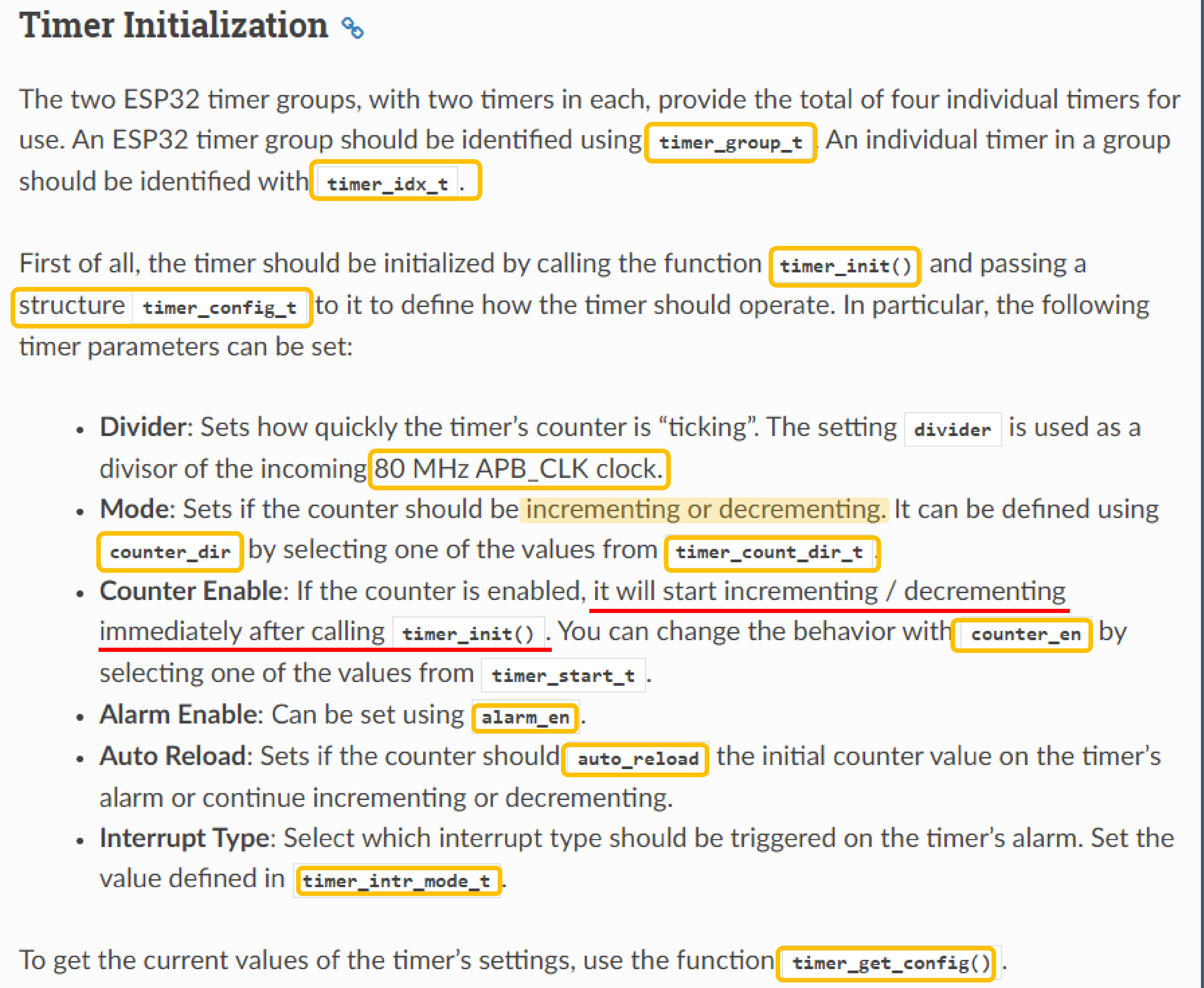
✓퀴즈 : 1초 마다 알람을 주는 나만의 타이머 만들어 보기



```c
/*
 * In this example, we will test hardware timer0 and timer1 of timer group0.
 */
void app_main(void)
{
    timer_queue = xQueueCreate(10, sizeof(timer_event_t));
    example_tg0_timer_init(TIMER_0, TEST_WITHOUT_RELOAD, TIMER_INTERVAL0_SEC);
    example_tg0_timer_init(TIMER_1, TEST_WITH_RELOAD,    TIMER_INTERVAL1_SEC);
    xTaskCreate(timer_example_evt_task, "timer_evt_task", 2048, NULL, 5, NULL);
}
```



```c
/**
 * @brief Selects a Timer-Group out of 2 available groups
 */
typedef enum {
    TIMER_GROUP_0 = 0, /*!<Hw timer group 0*/
    TIMER_GROUP_1 = 1, /*!<Hw timer group 1*/
    TIMER_GROUP_MAX,
} timer_group_t;

/**
 * @brief Select a hardware timer from timer groups
 */
typedef enum {
    TIMER_0 = 0, /*!<Select timer0 of GROUPx*/
    TIMER_1 = 1, /*!<Select timer1 of GROUPx*/
    TIMER_MAX,
} timer_idx_t;
```



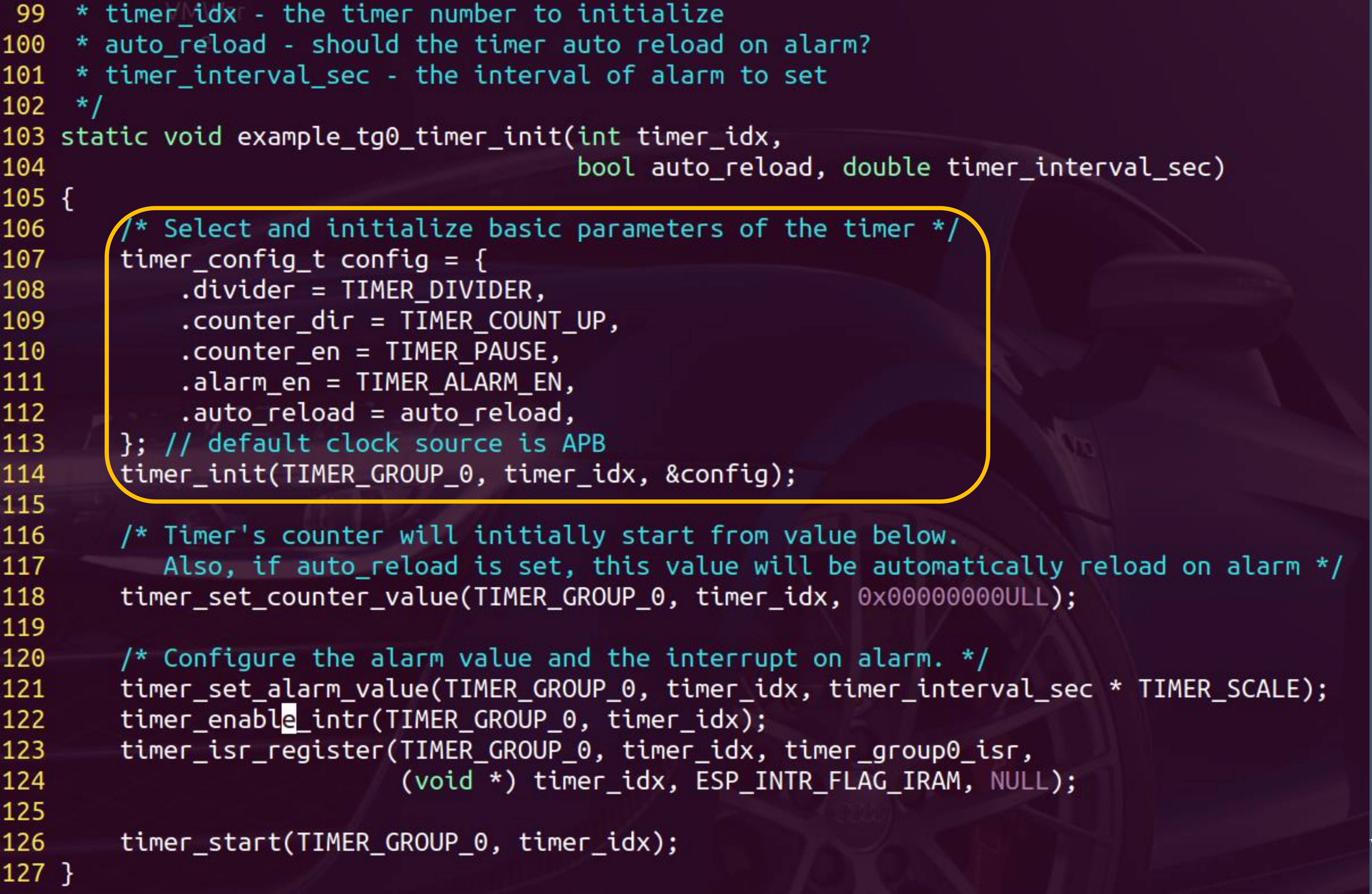
Timer Initialization

The two ESP32 timer groups, with two timers in each, provide the total of four individual timers for use. An ESP32 timer group should be identified using `timer_group_t`. An individual timer in a group should be identified with `timer_idx_t`.

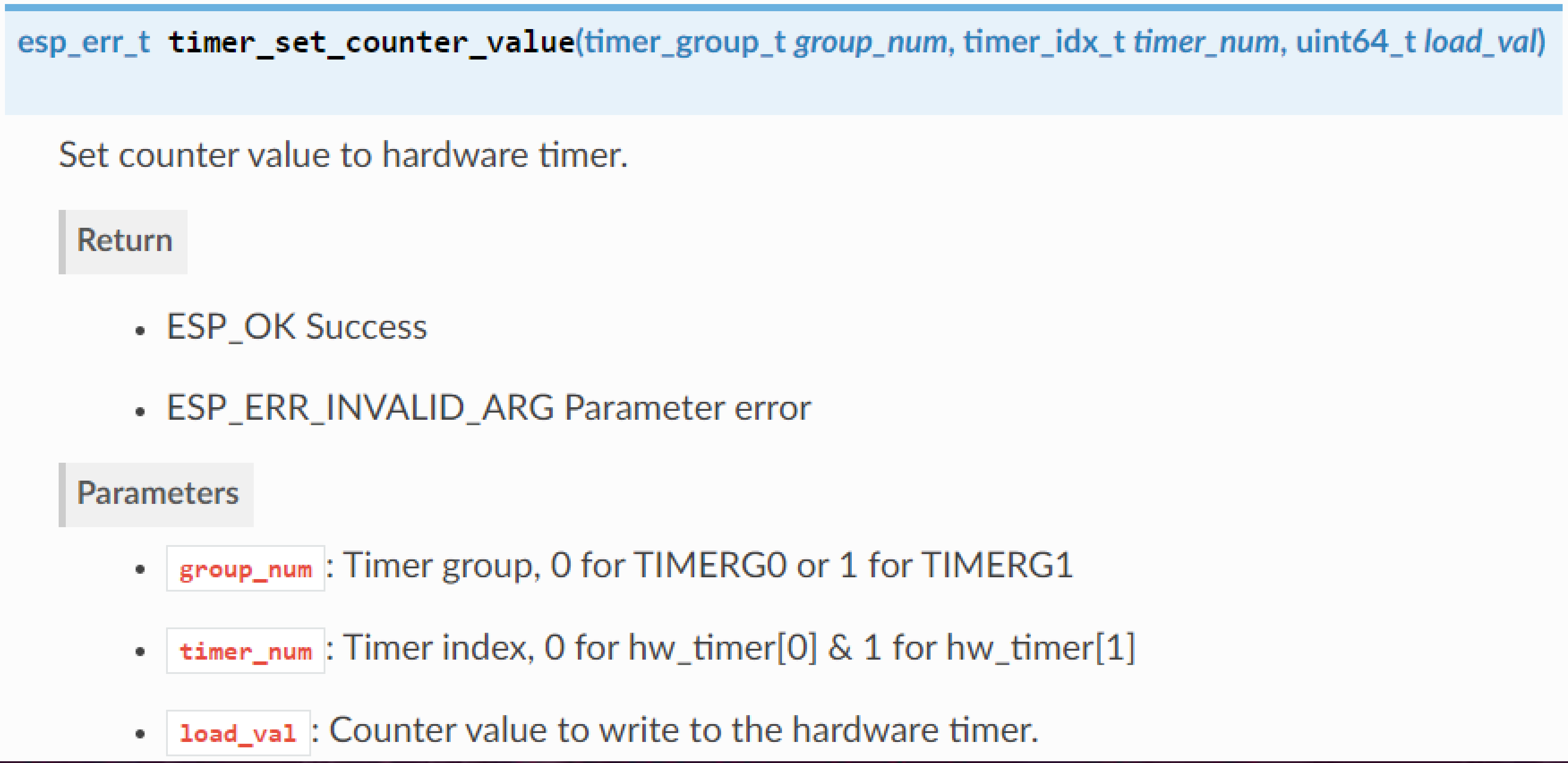
First of all, the timer should be initialized by calling the function `timer_init()` and passing a structure `timer_config_t` to it to define how the timer should operate. In particular, the following timer parameters can be set:

- **Divider**: Sets how quickly the timer's counter is "ticking". The setting `divider` is used as a divisor of the incoming 80 MHz APB_CLK clock.
- **Mode**: Sets if the counter should be incrementing or decrementing. It can be defined using `counter_dir` by selecting one of the values from `timer_count_dir_t`.
- **Counter Enable**: If the counter is enabled, it will start incrementing / decrementing immediately after calling `timer_init()`. You can change the behavior with `counter_en` by selecting one of the values from `timer_start_t`.
- **Alarm Enable**: Can be set using `alarm_en`.
- **Auto Reload**: Sets if the counter should `auto_reload` the initial counter value on the timer's alarm or continue incrementing or decrementing.
- **Interrupt Type**: Select which interrupt type should be triggered on the timer's alarm. Set the value defined in `timer_intr_mode_t`.

To get the current values of the timer's settings, use the function `timer_get_config()`.



```c
 * timer_idx - the timer number to initialize
 * auto_reload - should the timer auto reload on alarm?
 * timer_interval_sec - the interval of alarm to set
 */
static void example_tg0_timer_init(int timer_idx,
                                   bool auto_reload, double timer_interval_sec)
{
    /* Select and initialize basic parameters of the timer */
    timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_PAUSE,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = auto_reload,
    }; // default clock source is APB
    timer_init(TIMER_GROUP_0, timer_idx, &config);

    /* Timer's counter will initially start from value below.
       Also, if auto_reload is set, this value will be automatically reload on alarm */
    timer_set_counter_value(TIMER_GROUP_0, timer_idx, 0x00000000ULL);

    /* Configure the alarm value and the interrupt on alarm. */
    timer_set_alarm_value(TIMER_GROUP_0, timer_idx, timer_interval_sec * TIMER_SCALE);
    timer_enable_intr(TIMER_GROUP_0, timer_idx);
    timer_isr_register(TIMER_GROUP_0, timer_idx, timer_group0_isr,
                       (void *) timer_idx, ESP_INTR_FLAG_IRAM, NULL);

    timer_start(TIMER_GROUP_0, timer_idx);
}
```



esp_err_t **timer_set_counter_value**(timer_group_t *group_num*, timer_idx_t *timer_num*, uint64_t *load_val*)

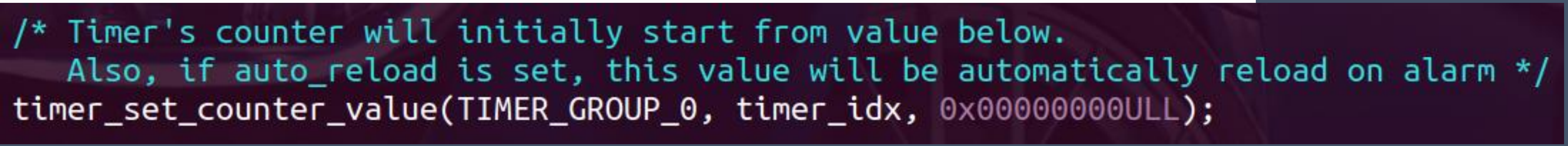
Set counter value to hardware timer.

Return

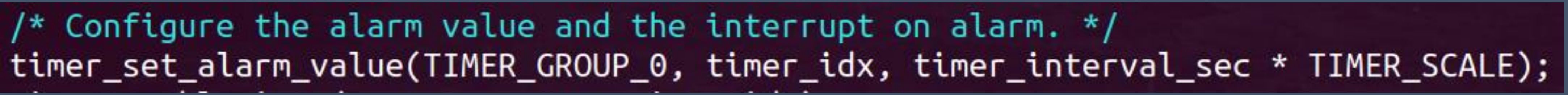
- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `group_num` : Timer group, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num` : Timer index, 0 for hw_timer[0] & 1 for hw_timer[1]
- `load_val` : Counter value to write to the hardware timer.

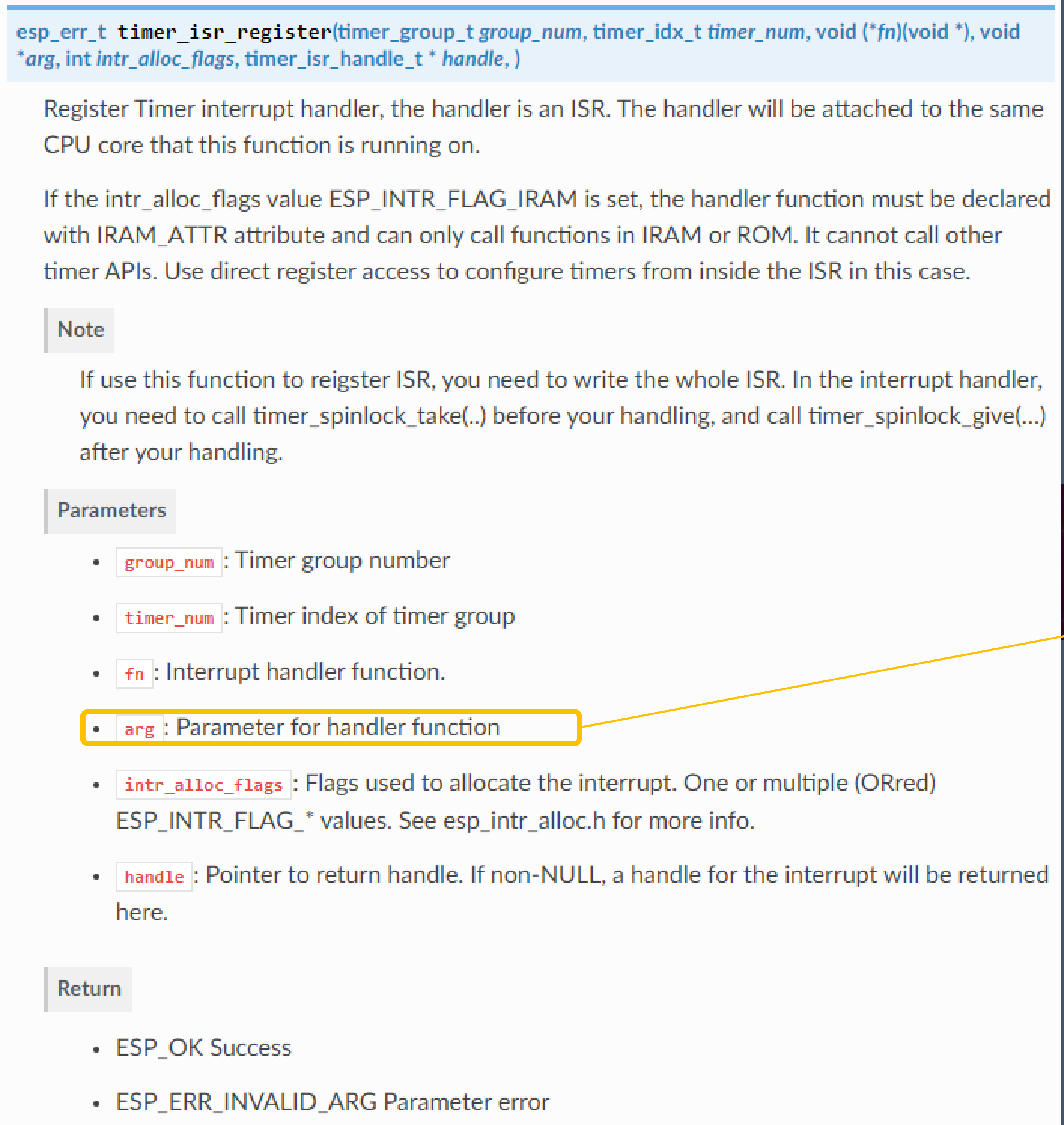


```
/* Timer's counter will initially start from value below.
   Also, if auto_reload is set, this value will be automatically reload on alarm */
timer_set_counter_value(TIMER_GROUP_0, timer_idx, 0x00000000ULL);
```



```
/* Configure the alarm value and the interrupt on alarm. */
timer_set_alarm_value(TIMER_GROUP_0, timer_idx, timer_interval_sec * TIMER_SCALE);
```

최초의 알람값 설정, 이후에는 ISR에서 인터럽트 발생시 재지정



esp_err_t **timer_isr_register**(timer_group_t *group_num*, timer_idx_t *timer_num*, void (**fn*)(void *), void **arg*, int *intr_alloc_flags*, timer_isr_handle_t * *handle*,)

Register Timer interrupt handler, the handler is an ISR. The handler will be attached to the same CPU core that this function is running on.

If the intr_alloc_flags value ESP_INTR_FLAG_IRAM is set, the handler function must be declared with IRAM_ATTR attribute and can only call functions in IRAM or ROM. It cannot call other timer APIs. Use direct register access to configure timers from inside the ISR in this case.

Note

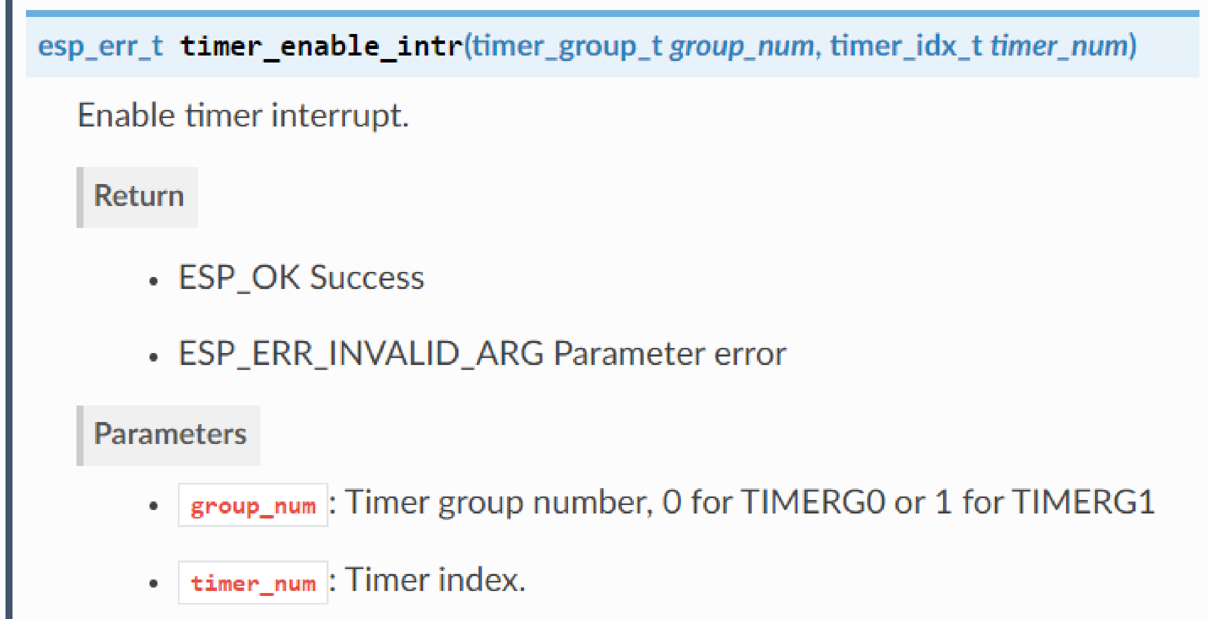
If use this function to reigster ISR, you need to write the whole ISR. In the interrupt handler, you need to call timer_spinlock_take(..) before your handling, and call timer_spinlock_give(...) after your handling.

Parameters

- `group_num`: Timer group number
- `timer_num`: Timer index of timer group
- `fn`: Interrupt handler function.
- `arg`: Parameter for handler function
- `intr_alloc_flags`: Flags used to allocate the interrupt. One or multiple (ORred) ESP_INTR_FLAG_* values. See esp_intr_alloc.h for more info.
- `handle`: Pointer to return handle. If non-NULL, a handle for the interrupt will be returned here.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error



esp_err_t **timer_enable_intr**(timer_group_t *group_num*, timer_idx_t *timer_num*)

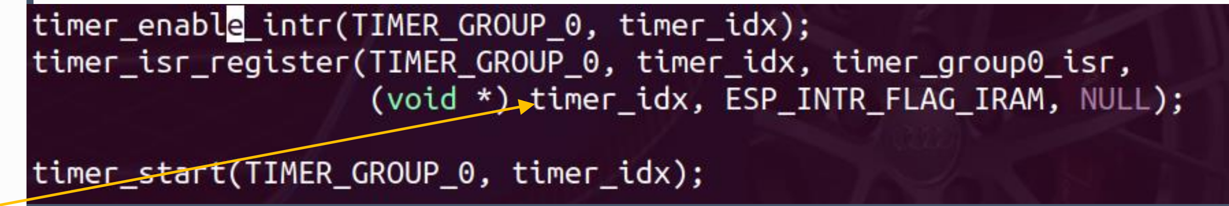
Enable timer interrupt.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

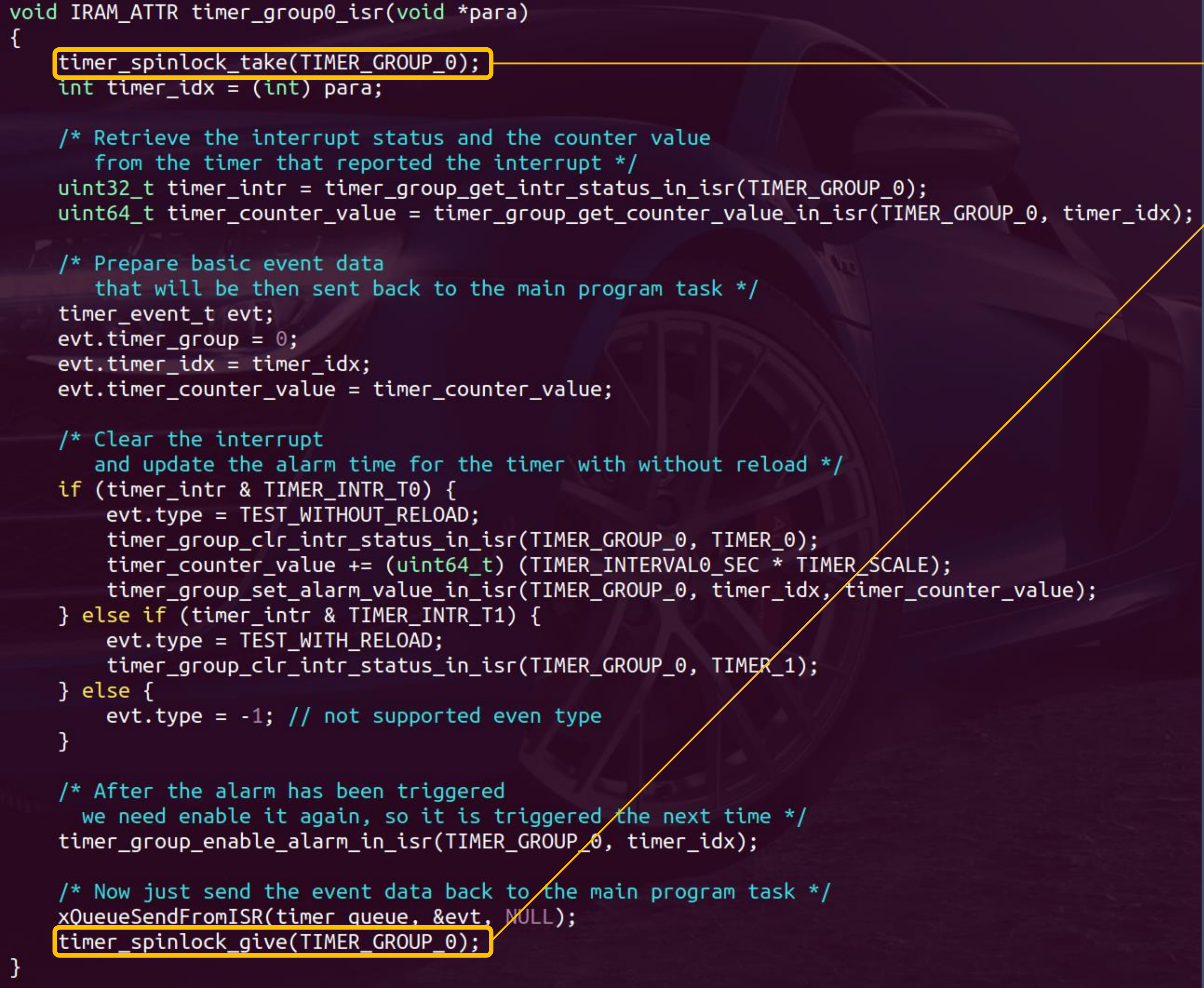
Parameters

- `group_num`: Timer group number, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num`: Timer index.



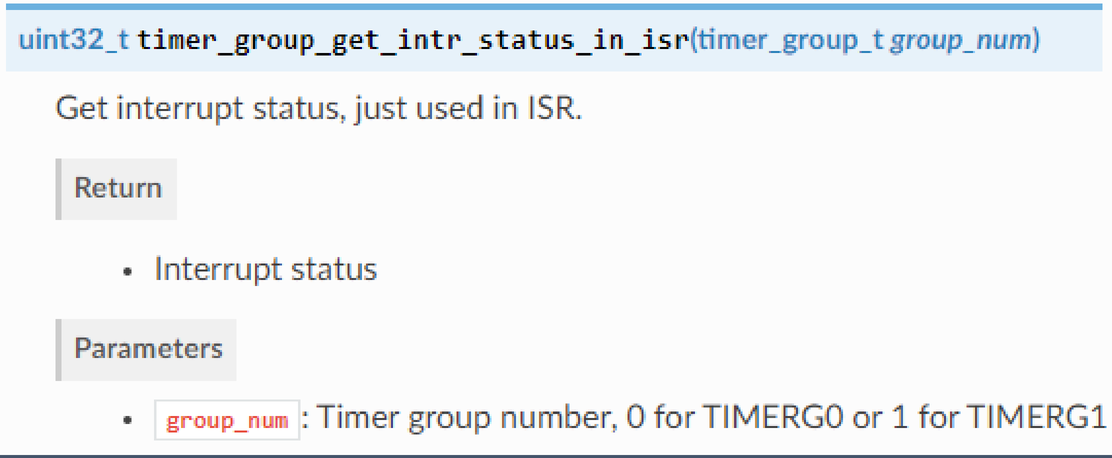
```
timer_enable_intr(TIMER_GROUP_0, timer_idx);
timer_isr_register(TIMER_GROUP_0, timer_idx, timer_group0_isr,
                   (void *) timer_idx, ESP_INTR_FLAG_IRAM, NULL);

timer_start(TIMER_GROUP_0, timer_idx);
```

www.CodeZoo.co.kr



```c
void IRAM_ATTR timer_group0_isr(void *para)
{
    timer_spinlock_take(TIMER_GROUP_0);
    int timer_idx = (int) para;

    /* Retrieve the interrupt status and the counter value
       from the timer that reported the interrupt */
    uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
    uint64_t timer_counter_value = timer_group_get_counter_value_in_isr(TIMER_GROUP_0, timer_idx);

    /* Prepare basic event data
       that will be then sent back to the main program task */
    timer_event_t evt;
    evt.timer_group = 0;
    evt.timer_idx = timer_idx;
    evt.timer_counter_value = timer_counter_value;

    /* Clear the interrupt
       and update the alarm time for the timer with without reload */
    if (timer_intr & TIMER_INTR_T0) {
        evt.type = TEST_WITHOUT_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
        timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
        timer_group_set_alarm_value_in_isr(TIMER_GROUP_0, timer_idx, timer_counter_value);
    } else if (timer_intr & TIMER_INTR_T1) {
        evt.type = TEST_WITH_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
    } else {
        evt.type = -1; // not supported even type
    }

    /* After the alarm has been triggered
      we need enable it again, so it is triggered the next time */
    timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);

    /* Now just send the event data back to the main program task */
    xQueueSendFromISR(timer_queue, &evt, NULL);
    timer_spinlock_give(TIMER_GROUP_0);
}
```

인터럽트 발생하면 처리할때까지
해당 타이머 그룹에 Lock을 건다



uint32_t **timer_group_get_intr_status_in_isr**(timer_group_t *group_num*)

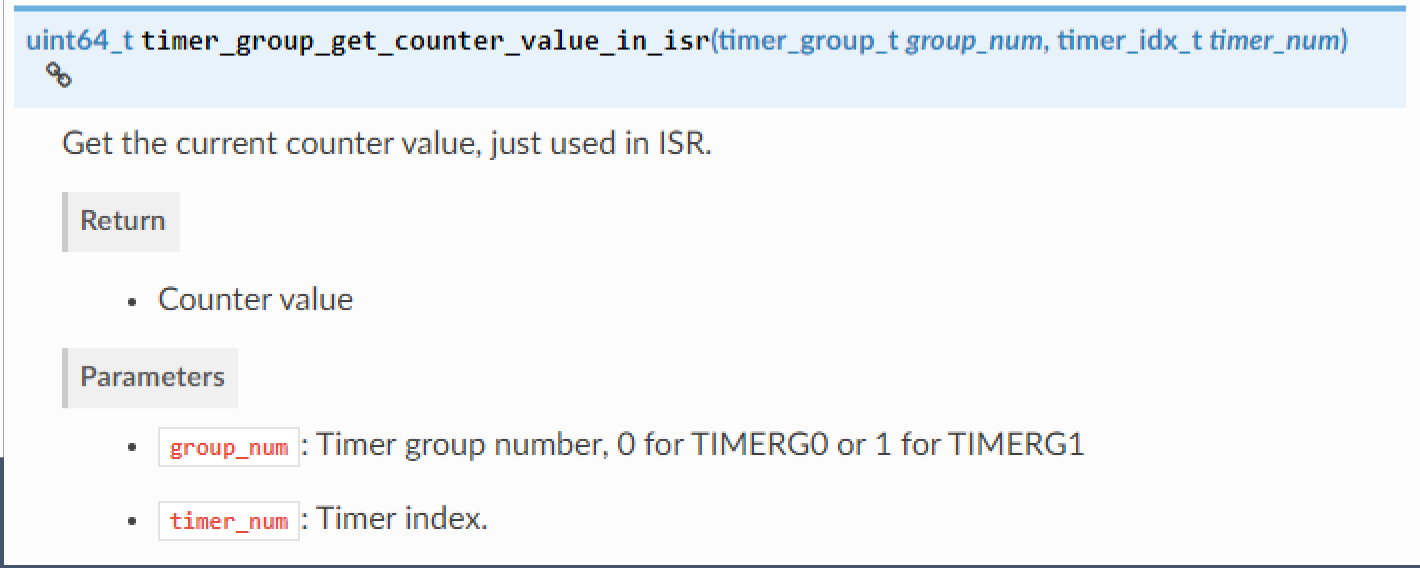
Get interrupt status, just used in ISR.

Return

- Interrupt status

Parameters

- `group_num` : Timer group number, 0 for TIMERG0 or 1 for TIMERG1



uint64_t **timer_group_get_counter_value_in_isr**(timer_group_t *group_num*, timer_idx_t *timer_num*)

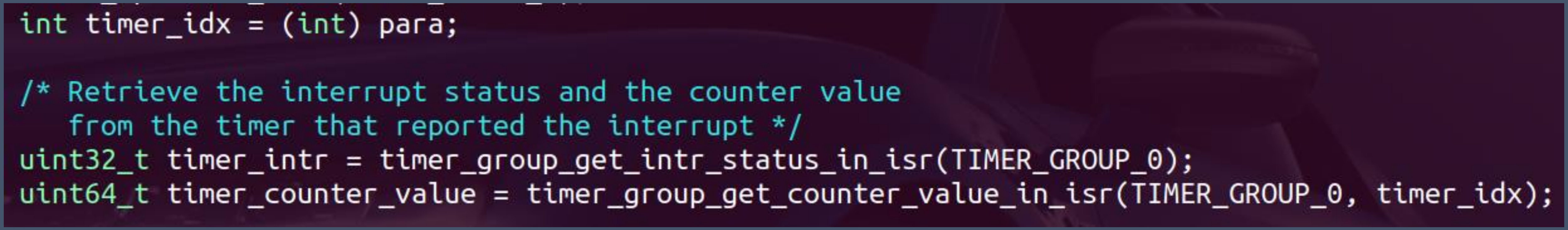
Get the current counter value, just used in ISR.

Return

- Counter value

Parameters

- `group_num` : Timer group number, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num` : Timer index.

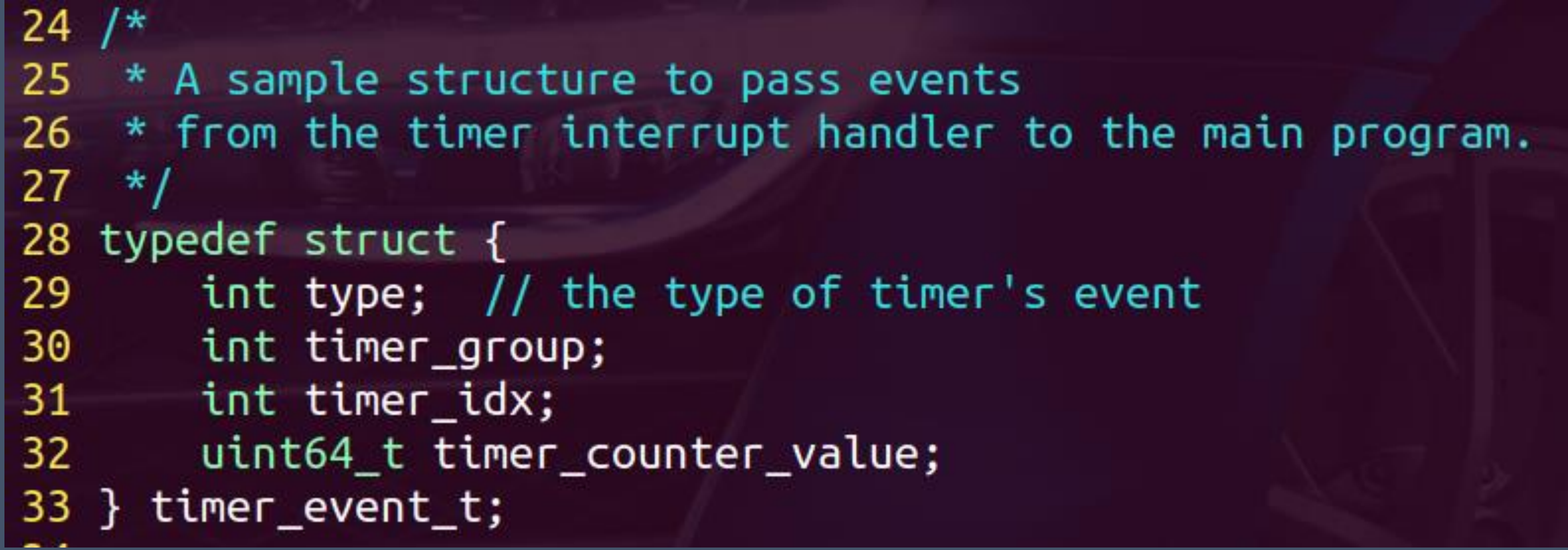


```
int timer_idx = (int) para;

/* Retrieve the interrupt status and the counter value
   from the timer that reported the interrupt */
uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
uint64_t timer_counter_value = timer_group_get_counter_value_in_isr(TIMER_GROUP_0, timer_idx);
```

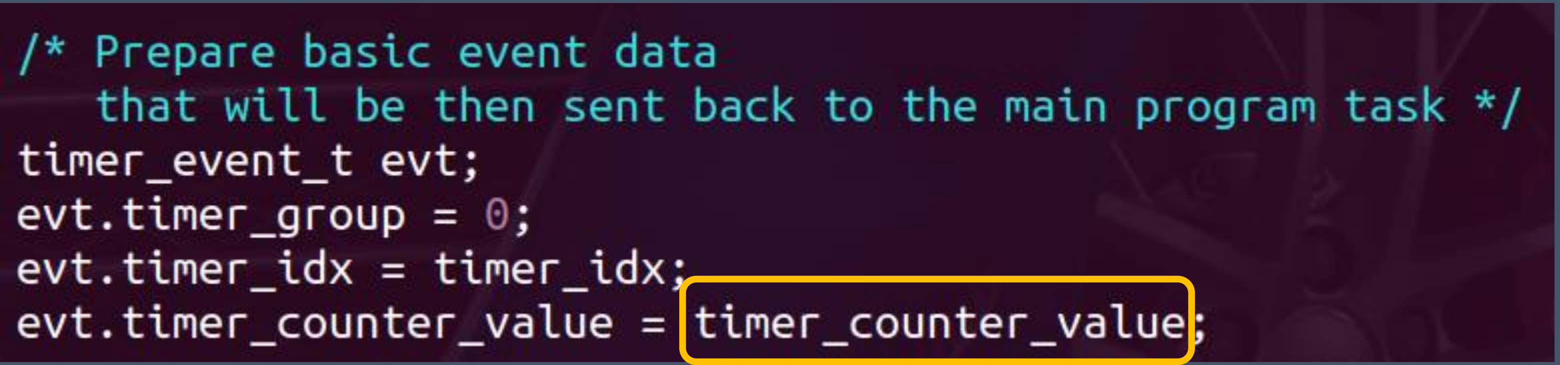
www.CodeZoo.co.kr

테스트코드에서 이벤트를 통해 상태를 전달하기 위해 임의로 만든 구조체

핵심이 아님!!!

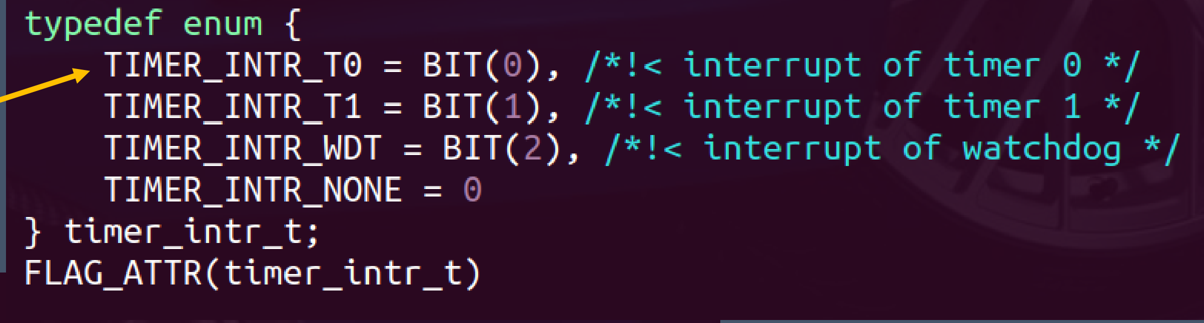


```
/*
 * A sample structure to pass events
 * from the timer interrupt handler to the main program.
 */
typedef struct {
    int type;  // the type of timer's event
    int timer_group;
    int timer_idx;
    uint64_t timer_counter_value;
} timer_event_t;
```

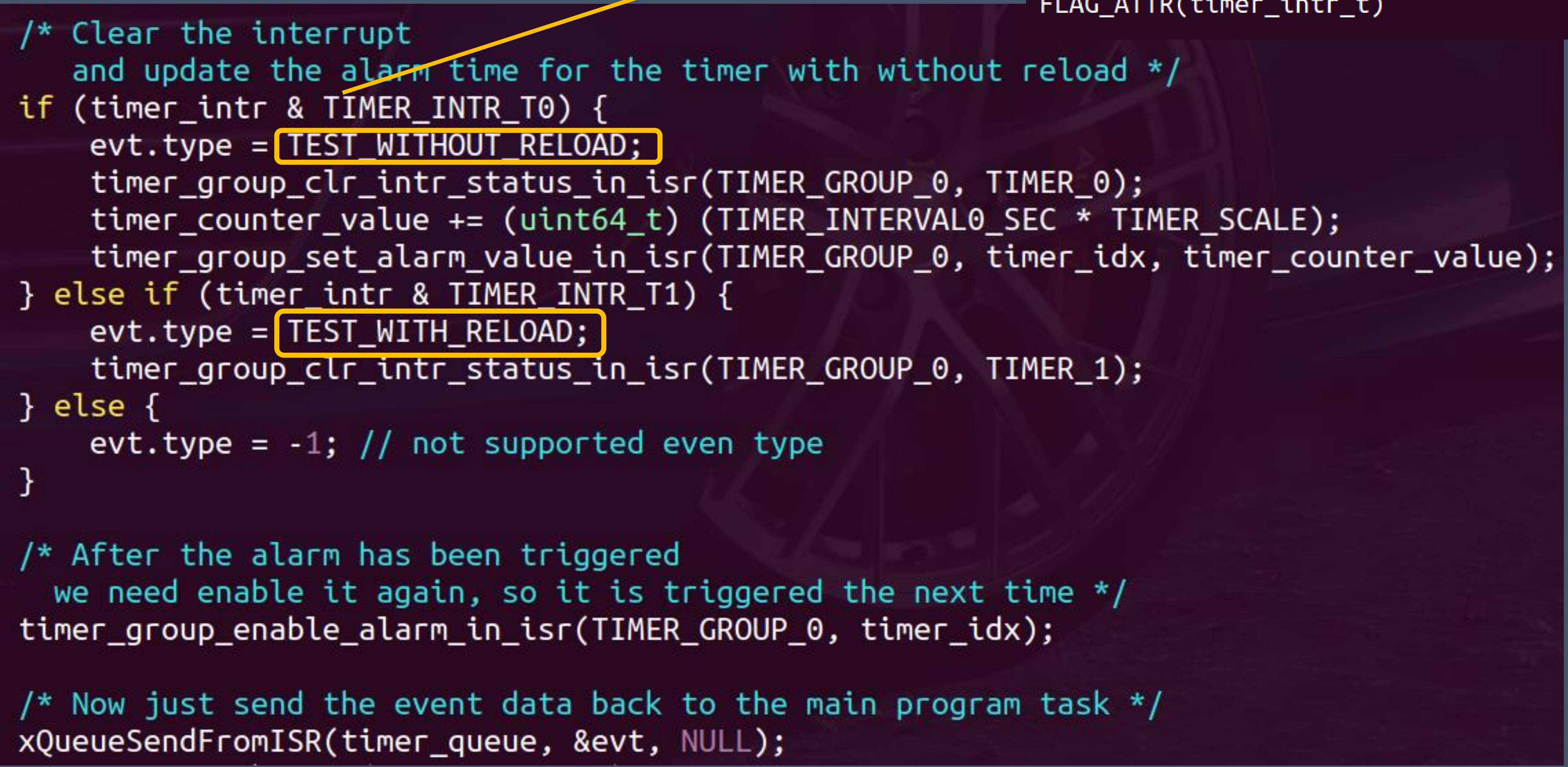


```
/* Prepare basic event data
   that will be then sent back to the main program task */
timer_event_t evt;
evt.timer_group = 0;
evt.timer_idx = timer_idx;
evt.timer_counter_value = timer_counter_value;
```

www.CodeZoo.co.kr

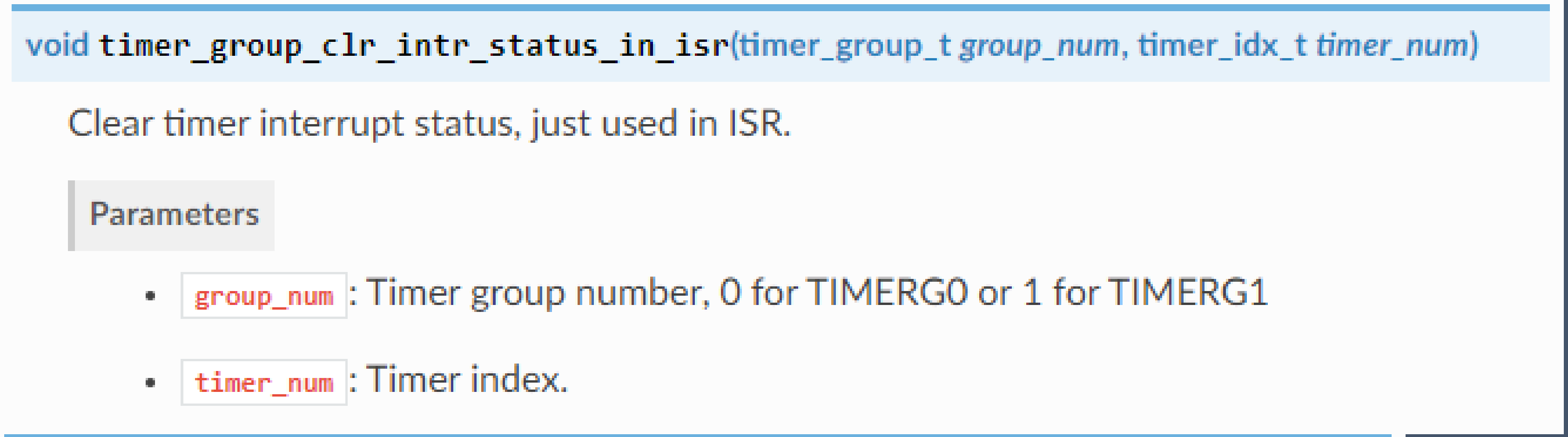


```
typedef enum {
    TIMER_INTR_T0 = BIT(0), /*!< interrupt of timer 0 */
    TIMER_INTR_T1 = BIT(1), /*!< interrupt of timer 1 */
    TIMER_INTR_WDT = BIT(2), /*!< interrupt of watchdog */
    TIMER_INTR_NONE = 0
} timer_intr_t;
FLAG_ATTR(timer_intr_t)
```



```
/* Clear the interrupt
   and update the alarm time for the timer with without reload */
if (timer_intr & TIMER_INTR_T0) {
    evt.type = TEST_WITHOUT_RELOAD;
    timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
    timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
    timer_group_set_alarm_value_in_isr(TIMER_GROUP_0, timer_idx, timer_counter_value);
} else if (timer_intr & TIMER_INTR_T1) {
    evt.type = TEST_WITH_RELOAD;
    timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
} else {
    evt.type = -1; // not supported even type
}

/* After the alarm has been triggered
  we need enable it again, so it is triggered the next time */
timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);

/* Now just send the event data back to the main program task */
xQueueSendFromISR(timer_queue, &evt, NULL);
```

RELOAD를 사용하지 않고 동일한 시간에 알람을 사용하려면 타이머 카운트 값을 내적해서 보관하고 다음 알람값을 더해서 timer_group_set_alarm_value_in_isr 로 지정해야 한다. (체크!!!)

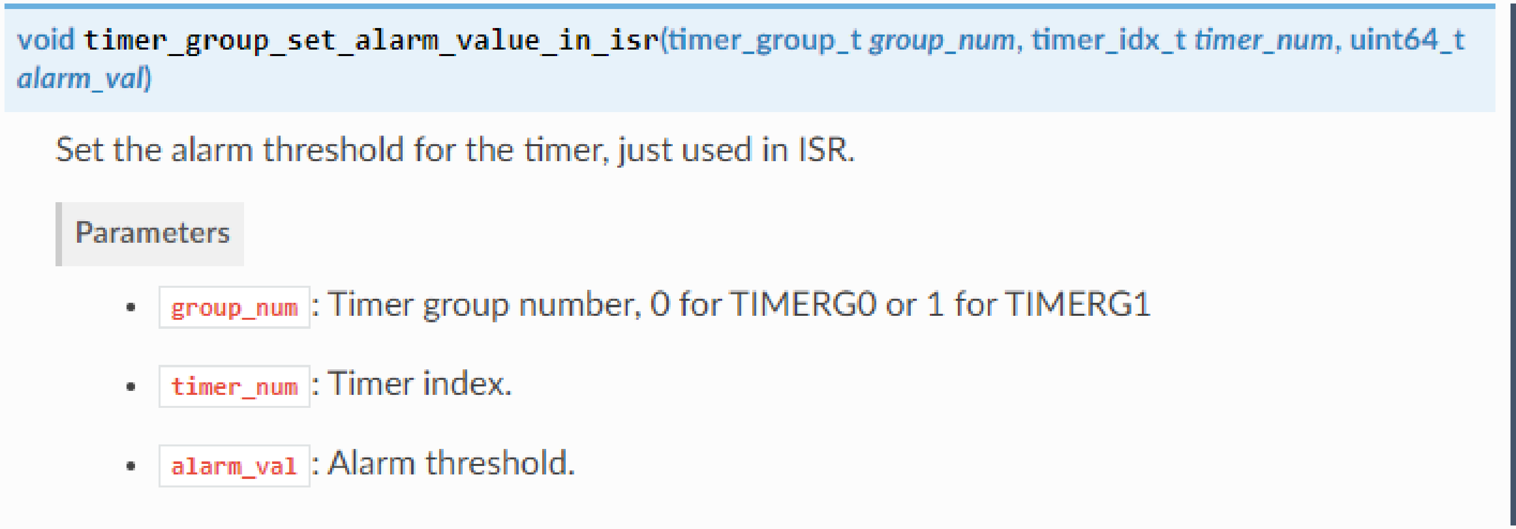


void `timer_group_clr_intr_status_in_isr`(timer_group_t *group_num*, timer_idx_t *timer_num*)

Clear timer interrupt status, just used in ISR.

Parameters

- `group_num` : Timer group number, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num` : Timer index.

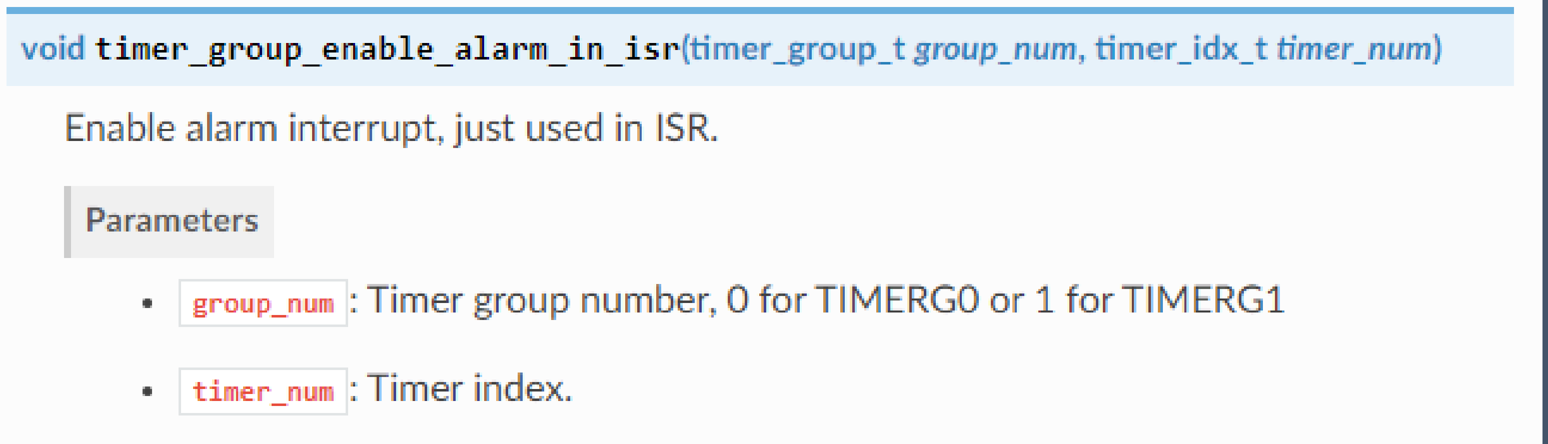


void `timer_group_set_alarm_value_in_isr`(timer_group_t *group_num*, timer_idx_t *timer_num*, uint64_t *alarm_val*)

Set the alarm threshold for the timer, just used in ISR.

Parameters

- `group_num` : Timer group number, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num` : Timer index.
- `alarm_val` : Alarm threshold.

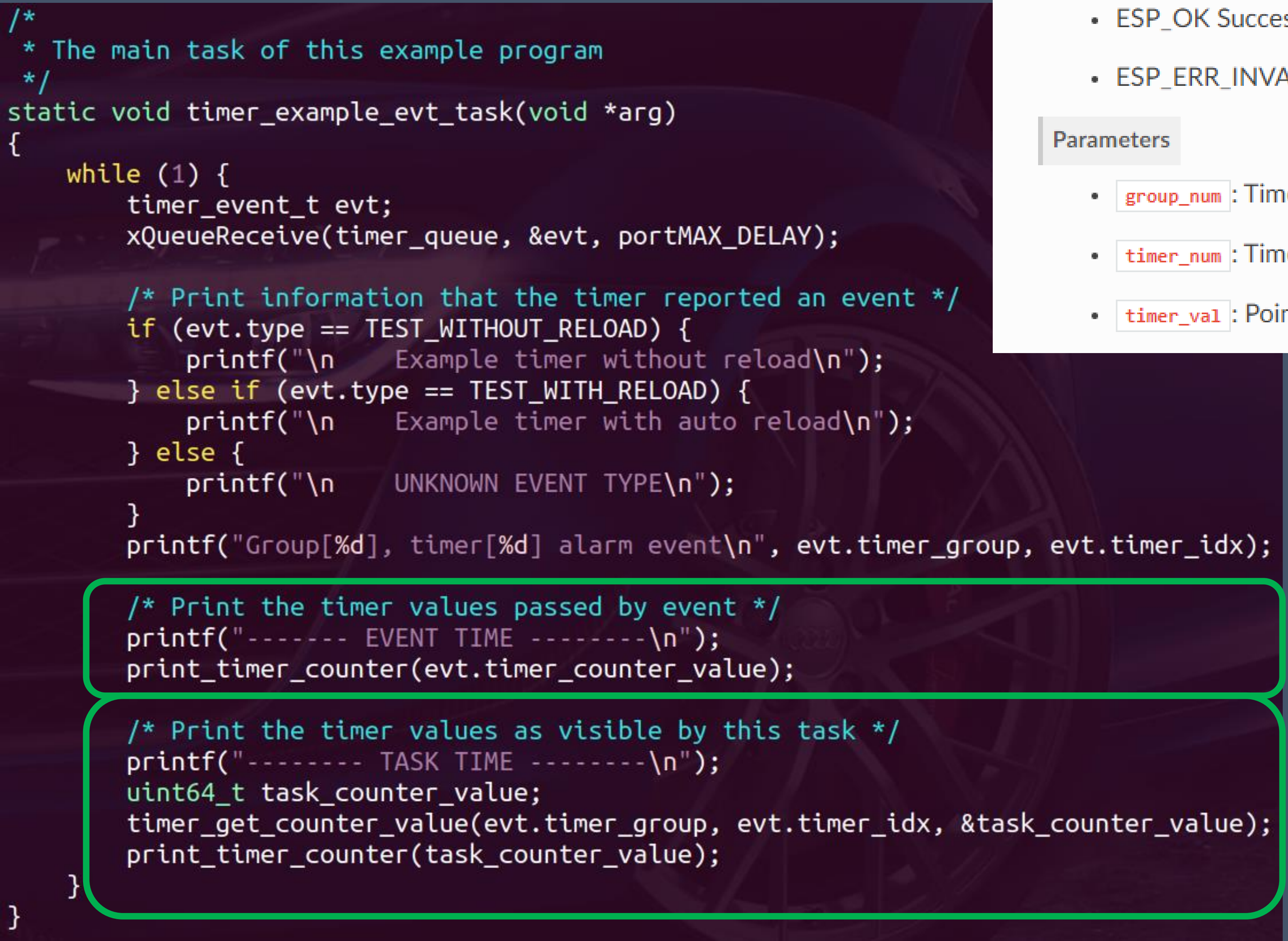


void `timer_group_enable_alarm_in_isr`(timer_group_t *group_num*, timer_idx_t *timer_num*)

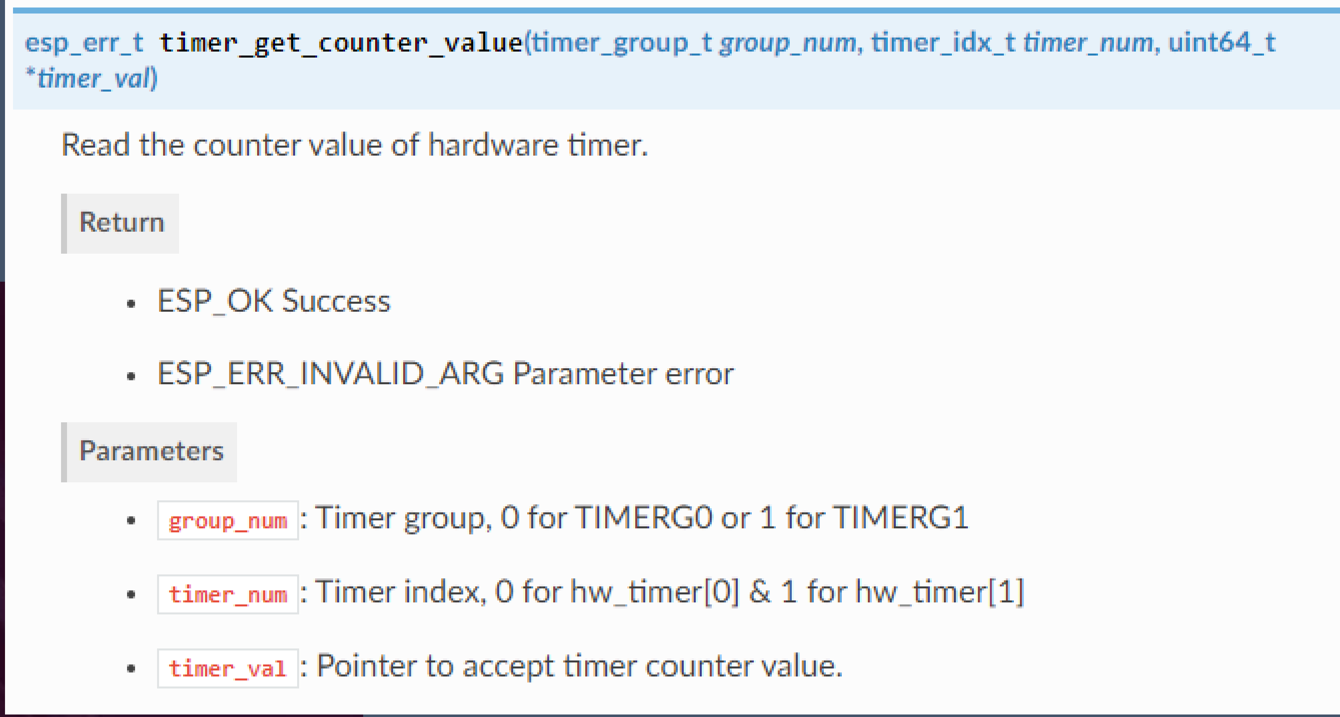
Enable alarm interrupt, just used in ISR.

Parameters

- `group_num` : Timer group number, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num` : Timer index.



```c
/*
 * The main task of this example program
 */
static void timer_example_evt_task(void *arg)
{
    while (1) {
        timer_event_t evt;
        xQueueReceive(timer_queue, &evt, portMAX_DELAY);

        /* Print information that the timer reported an event */
        if (evt.type == TEST_WITHOUT_RELOAD) {
            printf("\n    Example timer without reload\n");
        } else if (evt.type == TEST_WITH_RELOAD) {
            printf("\n    Example timer with auto reload\n");
        } else {
            printf("\n    UNKNOWN EVENT TYPE\n");
        }
        printf("Group[%d], timer[%d] alarm event\n", evt.timer_group, evt.timer_idx);

        /* Print the timer values passed by event */
        printf("------- EVENT TIME --------\n");
        print_timer_counter(evt.timer_counter_value);

        /* Print the timer values as visible by this task */
        printf("-------- TASK TIME --------\n");
        uint64_t task_counter_value;
        timer_get_counter_value(evt.timer_group, evt.timer_idx, &task_counter_value);
        print_timer_counter(task_counter_value);
    }
}
```



esp_err_t `timer_get_counter_value`(timer_group_t *group_num*, timer_idx_t *timer_num*, uint64_t **timer_val*)

Read the counter value of hardware timer.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `group_num`: Timer group, 0 for TIMERG0 or 1 for TIMERG1
- `timer_num`: Timer index, 0 for hw_timer[0] & 1 for hw_timer[1]
- `timer_val`: Pointer to accept timer counter value.

타이머가 동작 시간과
태스크까지 소비한 실제소비 시간을 확인

어제 GPIO 값을 읽어봤을때
0을 예상했는데, 1이 출력된 것과
연결해서 생각해 보기..

www.CodeZoo.co.kr

4. Analog to Digital Converter

✓ 2 개의 12 비트 SAR (Successive Approximation Register 연속 근사 레지스터) ADC가 포함된 총 18 개의 측정 채널 (아날로그 가능 핀)을 지원

✓ ADC 드라이버 API는 ADC1 (GPIO 32-39에 연결된 8 채널) 및 ADC2 (GPIO 0, 2, 4, 12-15 및 25-27에 연결된 10 채널) 지원. 그러나 ADC2 사용에는 응용 프로그램에 대한 몇 가지 제한 사항이 있음

- ADC2는 Wi-Fi 드라이버에서 사용됩니다. 따라서 응용 프로그램은 Wi-Fi 드라이버가 시작되지 않은 경우에만 ADC2를 사용할 수 있습니다.
- ADC2 핀 중 일부는 스트래핑 핀 (GPIO 0, 2, 15)으로 사용되므로 자유롭게 사용할 수 없습니다. 다음 공식 개발 키트의 경우에 해당합니다.

ESP32 DevKitC : GPIO 0은 외부 자동 프로그램 회로로 인해 사용할 수 없습니다.
ESP-WROVER-KIT : GPIO 0, 2, 4 및 15는 다른 목적으로 외부 연결로 인해 사용할 수 없습니다.

Configuration and Reading ADC

- ✓ 판독하기 전에 ADC를 구성해야 함

- ✓ ADC1를 들어, 구성은 함수를 호출하여 정밀도(precision)와 감쇠(attenuation)를 요구 adc1_config_width() and adc1_config_channel_atten().

- ✓ ADC2의 경우 adc2_config_channel_atten() 로 감쇠를 구성. ADC2의 판독 폭은 판독 할 때마다 구성 됨

- ✓ adc1_channel_t 과 adc2_channel_t 로 감쇠 구성 참조 채널별로 수행. 상기 함수의 파라미터 설정

- ✓ adc1_get_raw(), adc2_get_raw() 로 ADC 변환 결과를 읽을 수 있음. ADC2의 판독 폭은 adc2_get_raw() 대신에 파라미터로 설정해야 함

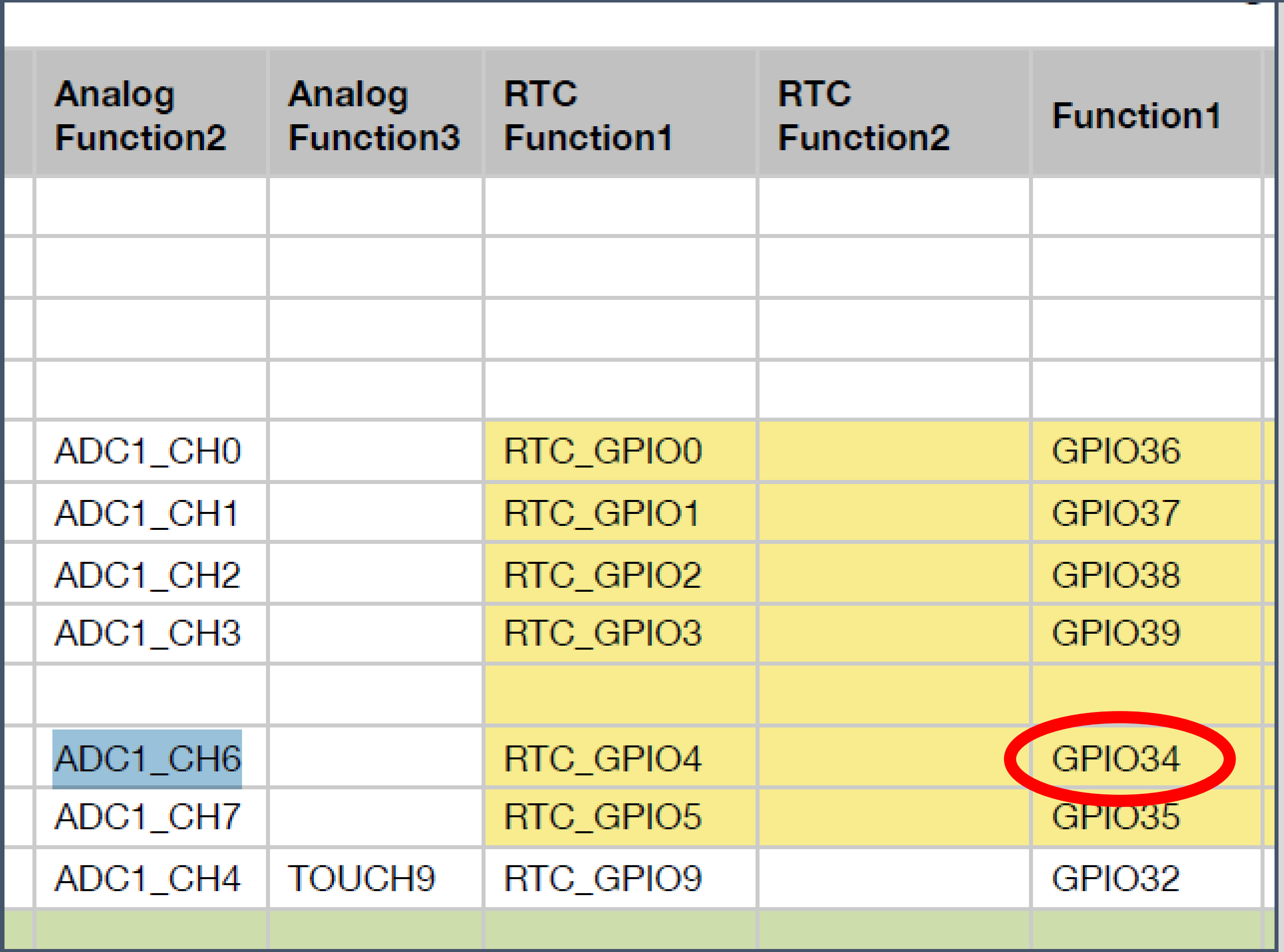
노트

- ✓ ADC2는 판독 동작, 높은 우선 순위를 갖는 무선 랜 모듈과 공유되기 때문에 esp_wifi_start()와 esp_wifi_stop() 사이에 실패 할수 있음. 리턴 코드를 사용하여 읽기가 성공적인지 확인해야 함.

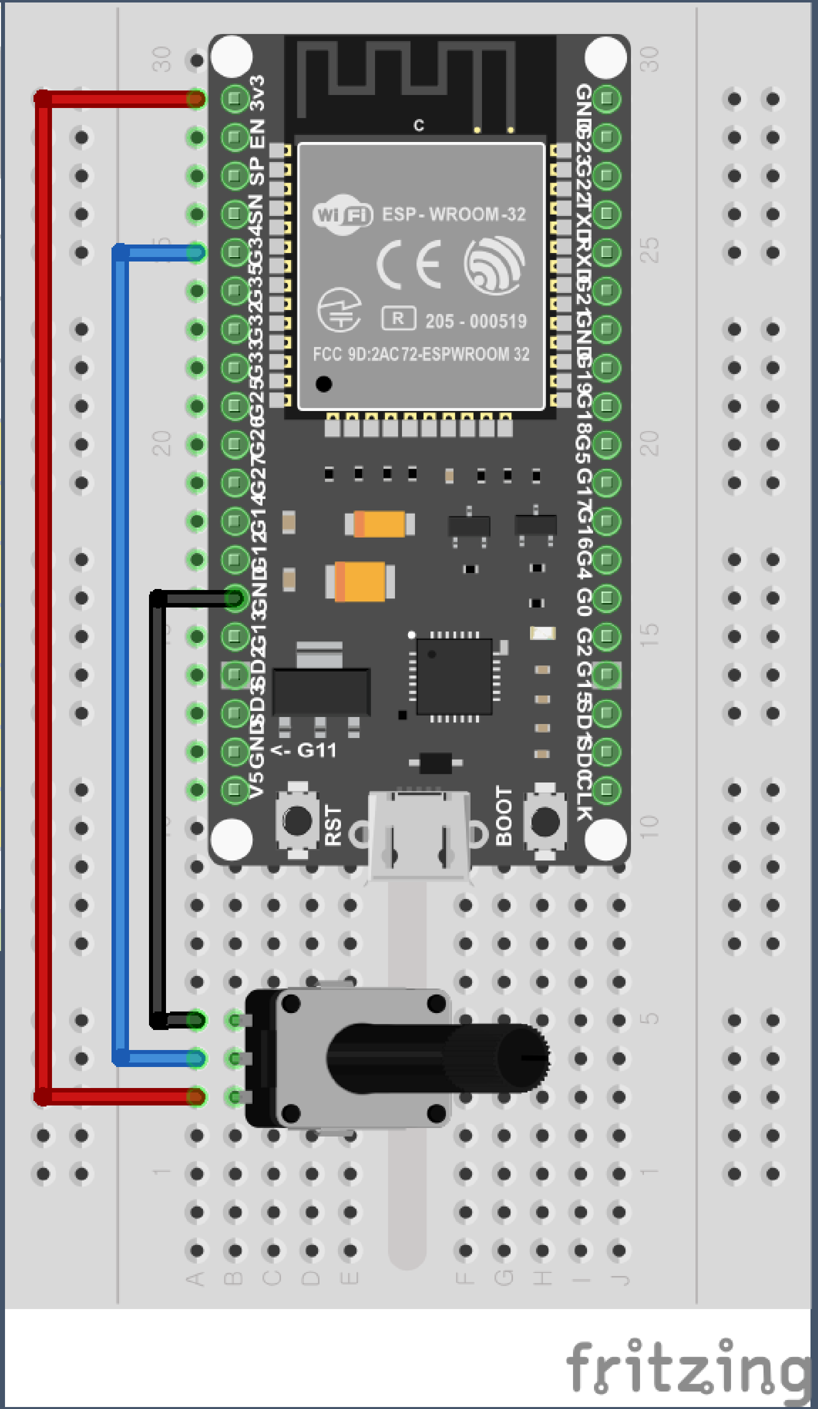
- ✓ 전용 기능을 호출하여 ADC1을 통해 hall_sensor_read() 로 내부 홀 센서를 읽을 수도 있음. 홀 센서도 ESP32 내부에 있으며 ADC1 (GPIO 36 및 39)의 채널 0 및 3을 사용합니다. 이 핀에 다른 것을 연결하지 말고 구성을 변경하지 마십시오. 그렇지 않으면 센서의 낮은 값 신호 측정에 영향을 줄 수 있음.

- ✓ 이 API는 ULP 에서 읽도록 ADC1을 구성하는 편리한 방법을 제공. 이렇게 하려면 함수를 호출 adc1_ulp_enable()한 다음 위에서 설명한대로 정밀도와 감쇠를 설정

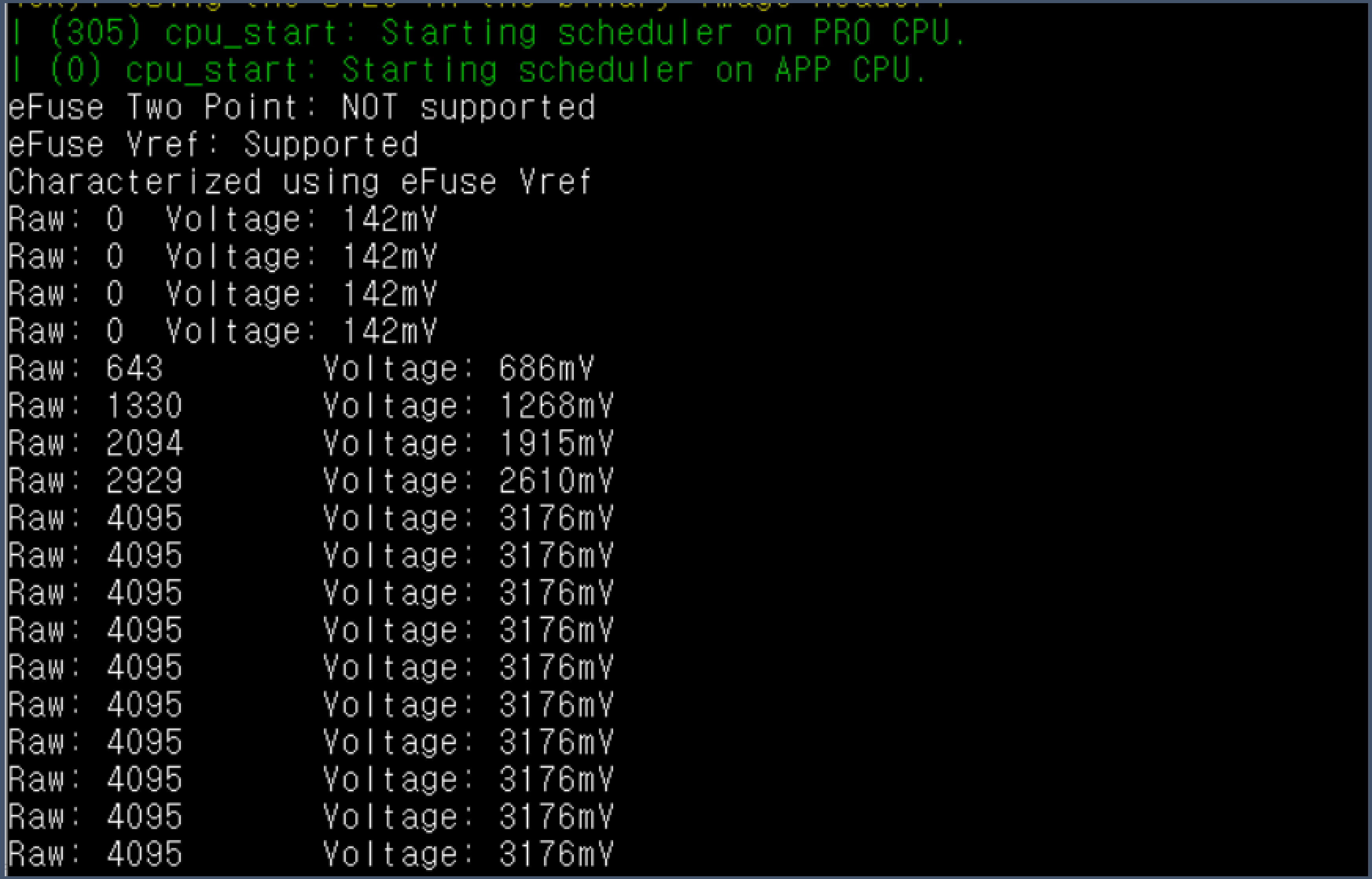
- ✓ adc2_vref_to_gpio()내부 기준 전압을 GPIO 핀으로 라우팅하는 데 사용되는 또 다른 특정 기능이 있습니다 . ADC 판독을 교정하는 것이 편리하며 이는 노이즈 최소화 섹션에서 설명. (Minimizing Noise)



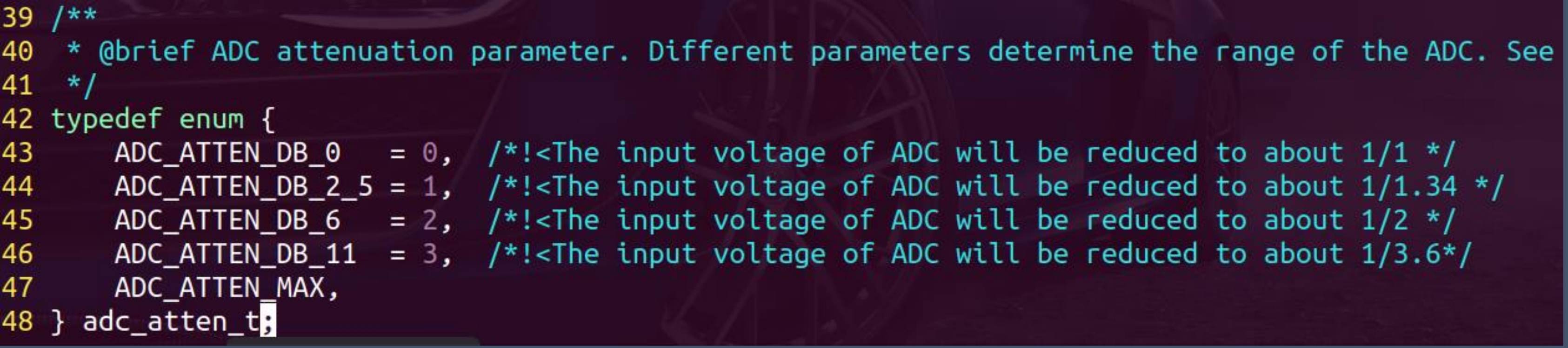
Analog Function2	Analog Function3	RTC Function1	RTC Function2	Function1
ADC1_CH0		RTC_GPIO0		GPIO36
ADC1_CH1		RTC_GPIO1		GPIO37
ADC1_CH2		RTC_GPIO2		GPIO38
ADC1_CH3		RTC_GPIO3		GPIO39
ADC1_CH6		RTC_GPIO4		GPIO34
ADC1_CH7		RTC_GPIO5		GPIO35
ADC1_CH4	TOUCH9	RTC_GPIO9		GPIO32



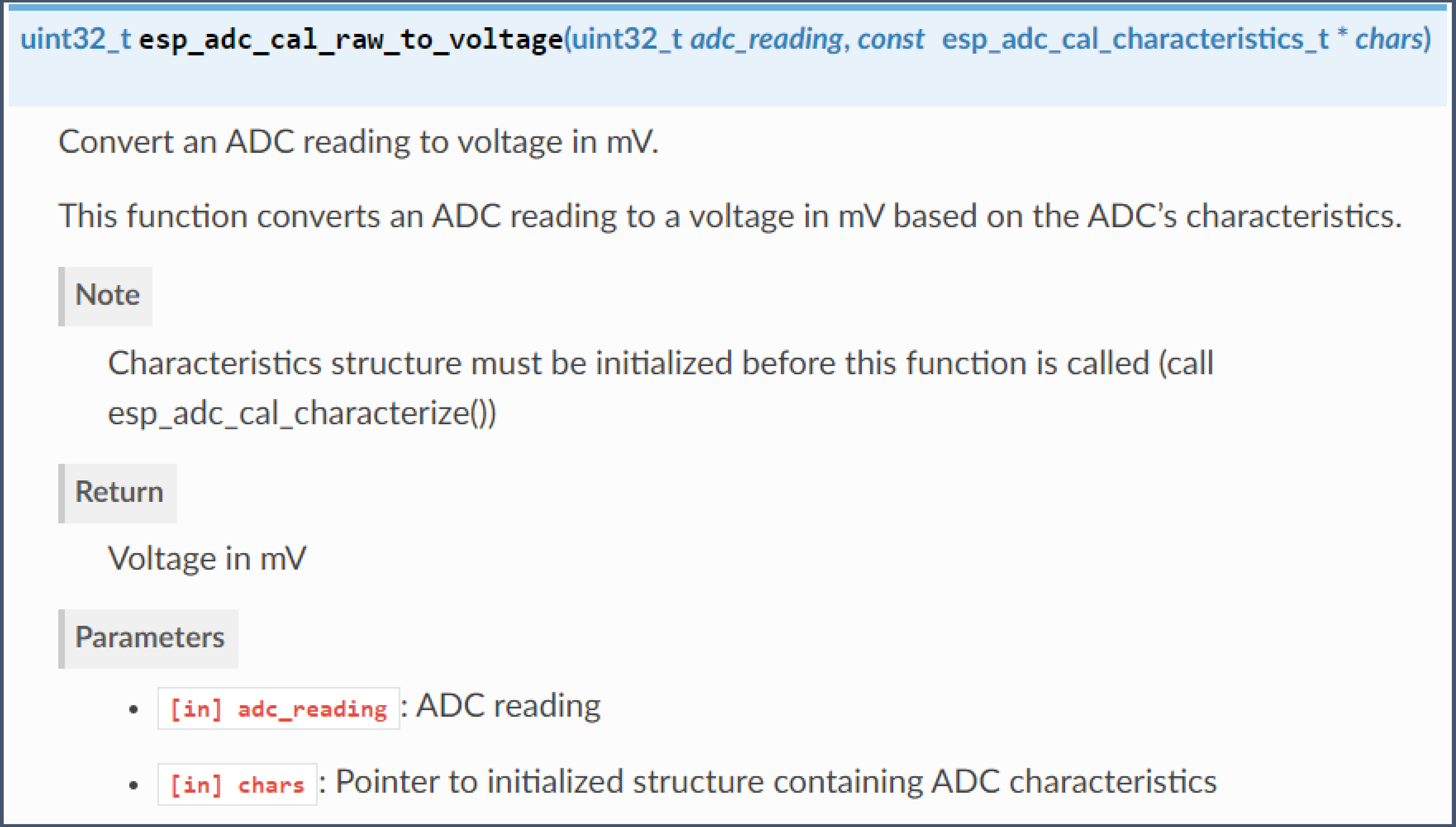
www.CodeZoo.co.kr



```
I (305) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
eFuse Two Point: NOT supported
eFuse Vref: Supported
Characterized using eFuse Vref
Raw: 0  Voltage: 142mV
Raw: 0  Voltage: 142mV
Raw: 0  Voltage: 142mV
Raw: 0  Voltage: 142mV
Raw: 643        Voltage: 686mV
Raw: 1330       Voltage: 1268mV
Raw: 2094       Voltage: 1915mV
Raw: 2929       Voltage: 2610mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
Raw: 4095       Voltage: 3176mV
```



```
/**
 * @brief ADC attenuation parameter. Different parameters determine the range of the ADC. See
 */
typedef enum {
    ADC_ATTEN_DB_0   = 0,  /*!<The input voltage of ADC will be reduced to about 1/1 */
    ADC_ATTEN_DB_2_5 = 1,  /*!<The input voltage of ADC will be reduced to about 1/1.34 */
    ADC_ATTEN_DB_6   = 2,  /*!<The input voltage of ADC will be reduced to about 1/2 */
    ADC_ATTEN_DB_11  = 3,  /*!<The input voltage of ADC will be reduced to about 1/3.6*/
    ADC_ATTEN_MAX,
} adc_atten_t;
```



uint32_t **esp_adc_cal_raw_to_voltage**(uint32_t *adc_reading*, const esp_adc_cal_characteristics_t * *chars*)

Convert an ADC reading to voltage in mV.

This function converts an ADC reading to a voltage in mV based on the ADC's characteristics.

Note

Characteristics structure must be initialized before this function is called (call esp_adc_cal_characterize())

Return

Voltage in mV

Parameters

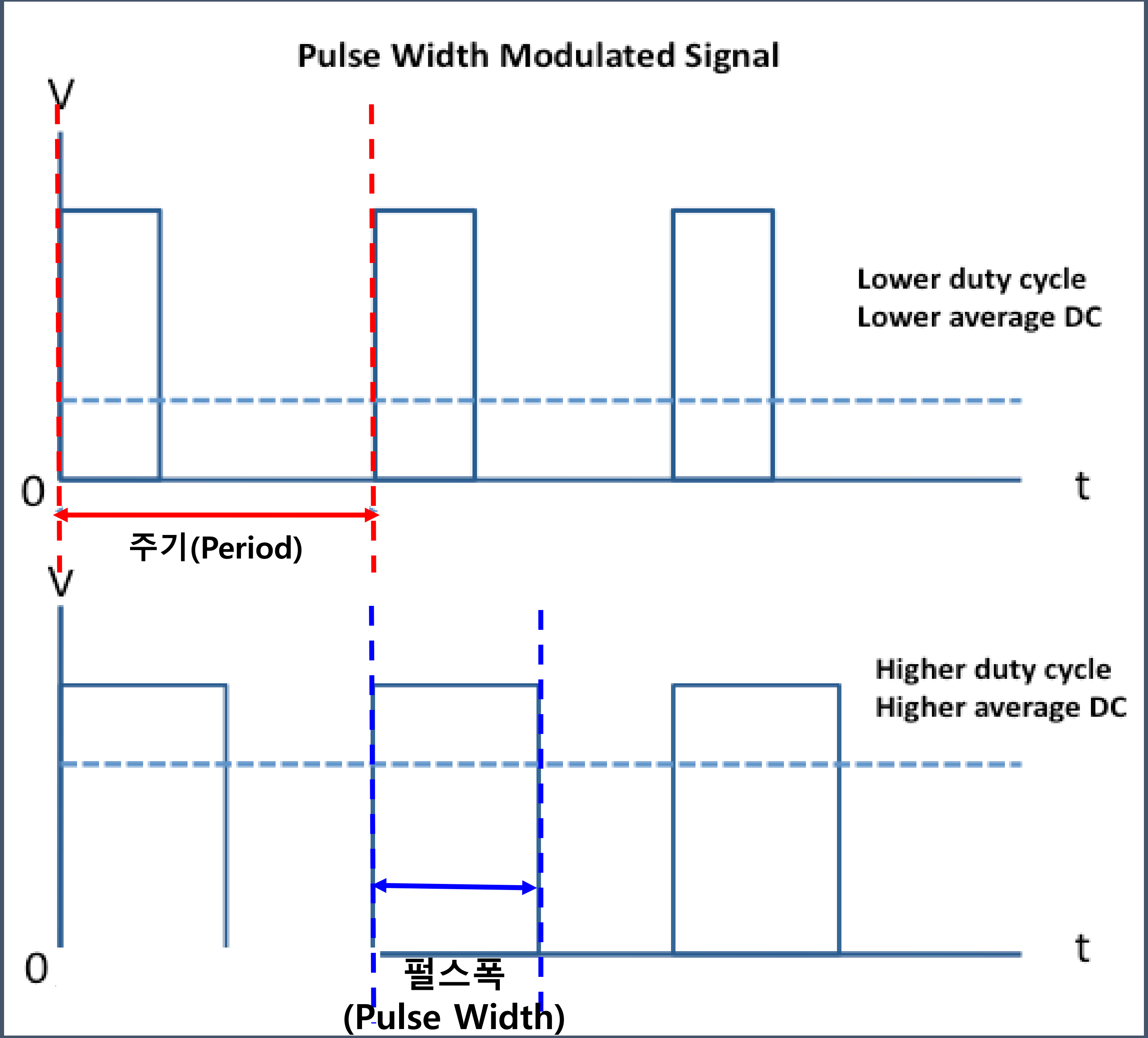
- `[in] adc_reading` : ADC reading
- `[in] chars` : Pointer to initialized structure containing ADC characteristics

www.CodeZoo.co.kr

PWM

PWM(Pulse Width Modulation) 이란?
PWM은 한국말로 '펄스 폭 변조'를 말합니다.
펄스 폭 변조란 아래 그림과 같이 사각파 펄스 폭을 조절하면 출력되는 전압을 조절할수 있게 됩니다
예를 들어 0~5V까지 출력이 가능한 PWM에서 2.5V의 출력을 만든다고 한다면
PWM 출력에서 0V와 5V의 변화를 빠른 주기(Period)로 같은 펄스폭(pulse width)으로
(0505050...)바꿔주는데 주기에 비례한 펄스폭이 50%를 유지한다면 출력 전압은 2.5V인 것처럼 인식하게 됩니다.
이렇게 PWM의 펄스폭의 변화를 주면 원하는 아날로그 전압을 디지털 출력을 만들수 있습니다.

* 펄스폭의 비율을 "Duty Ratio" 또는 "Duty cycle"라고 합니다.

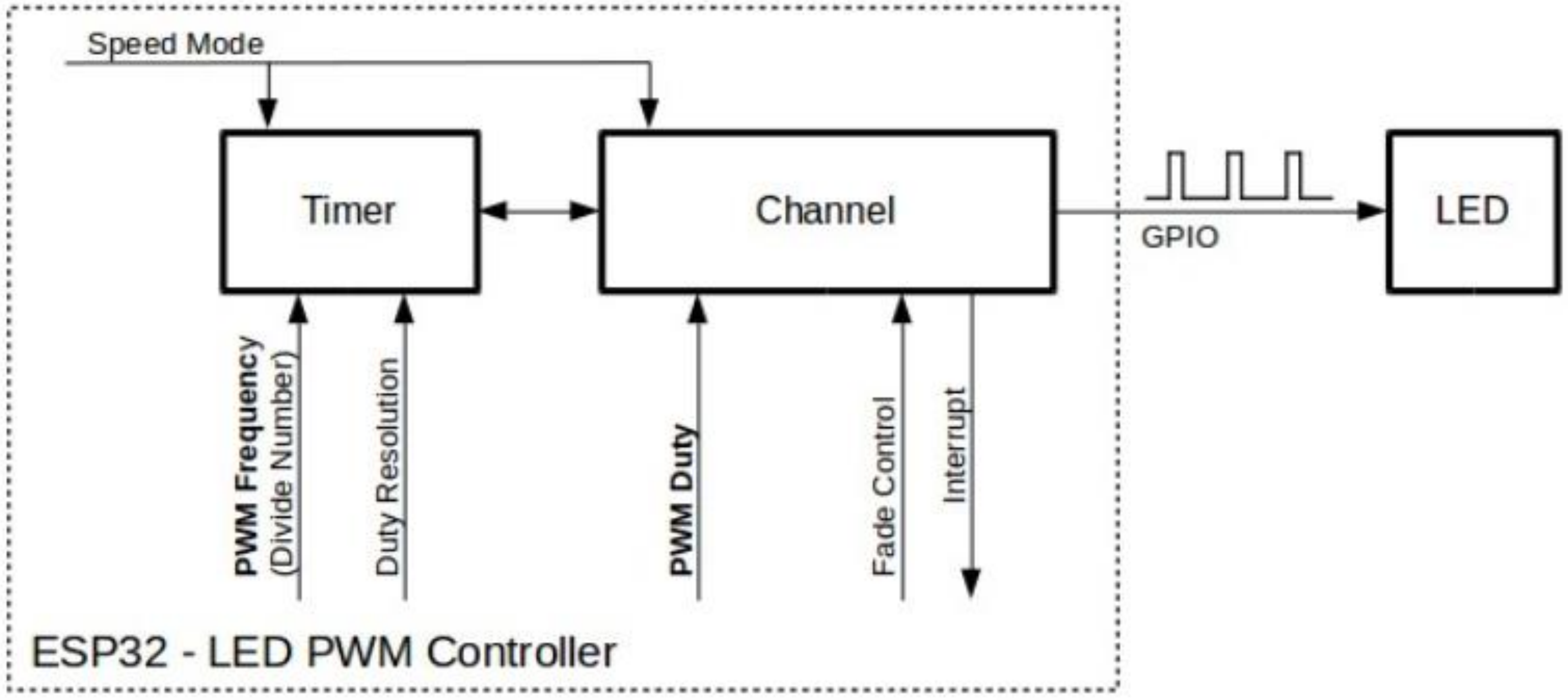


5. PWM (LED Control)

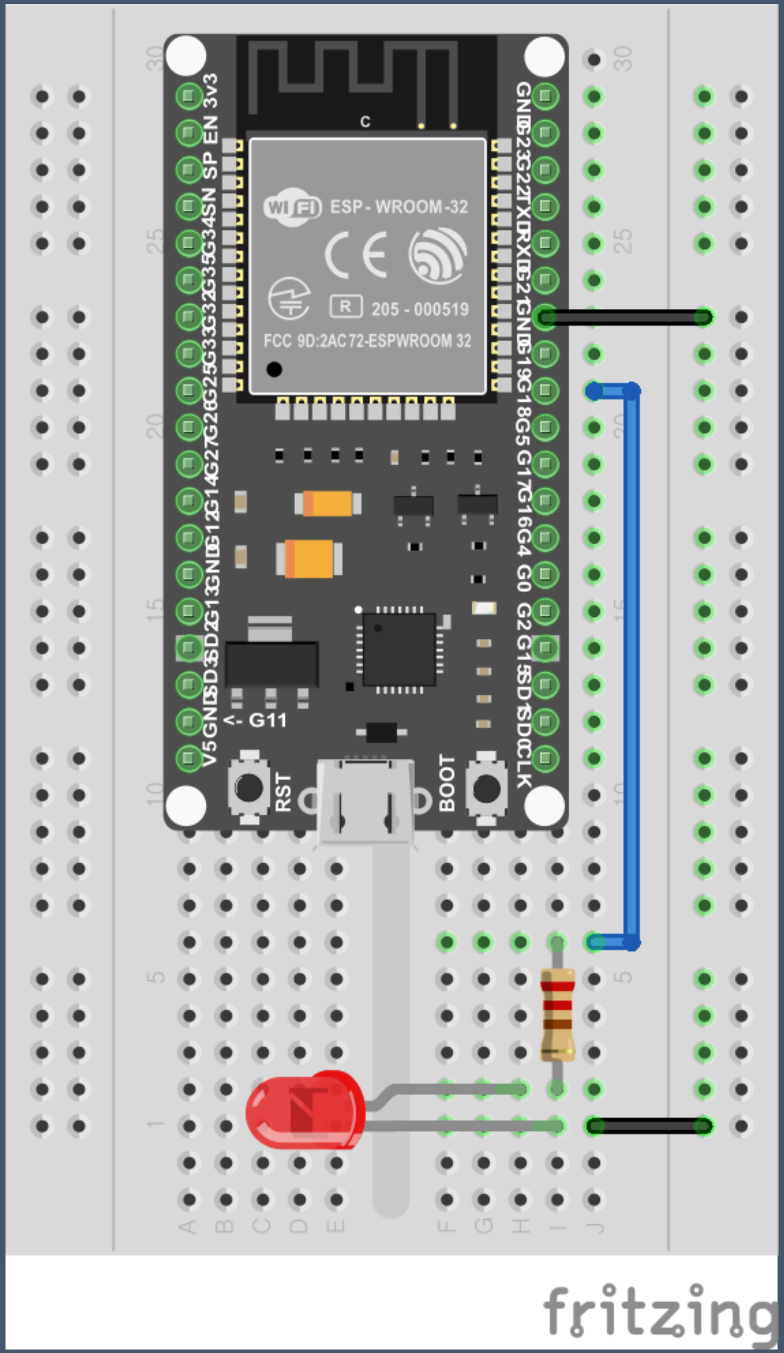
- ✓ LED 제어 (LEDC) 주변 장치는 주로 LED의 강도를 제어하도록 설계되었지만 다른 목적으로도 PWM 신호를 생성하는 데에도 사용할 수 있습니다. 예를 들어 RGB LED 장치를 구동하는 데 사용할 수있는 독립적인 파형을 생성 할 수 있는 16 개의 채널이 있습니다.

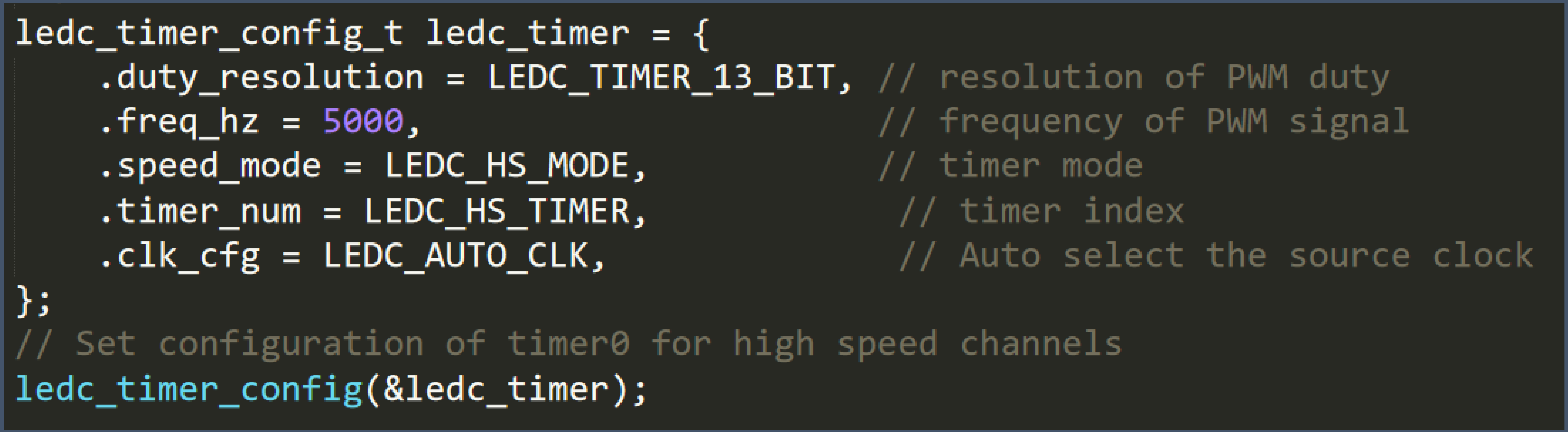
- ✓ LEDC 채널은 각각 8 채널의 두 그룹으로 나뉩니다. 한 그룹의 LEDC 채널은 고속 모드에서 작동합니다. 이 모드는 하드웨어로 구현되며 PWM 듀티 사이클을 자동으로 글리치 없이(glitch-free) 변경할 수 있습니다. 다른 채널 그룹은 저속 모드로 작동하므로 소프트웨어의 드라이버가 PWM 듀티 사이클을 변경해야합니다. 각 채널 그룹은 다른 클럭 소스를 사용할 수도 있습니다.

- ✓ PWM 컨트롤러는 듀티 사이클을 점진적으로 증가 또는 감소시켜 프로세서 간섭없이 페이드를 허용합니다.

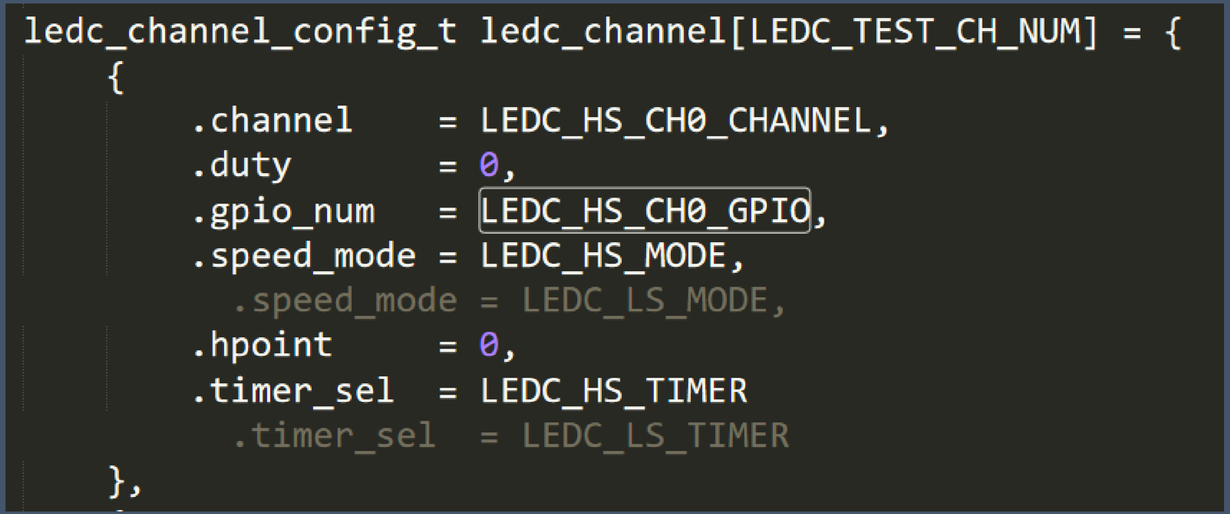


LED PWM 컨트롤러 API의 주요 설정

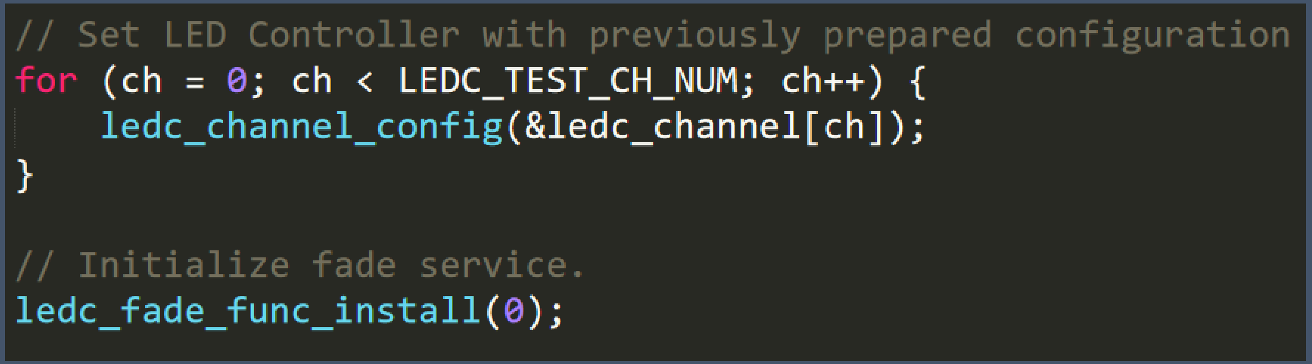

ESP-WROOM-32
205 - 000519
FCC 9D:2AC72-ESPWROOM 32
RST
BOOT
<- G11
fritzing



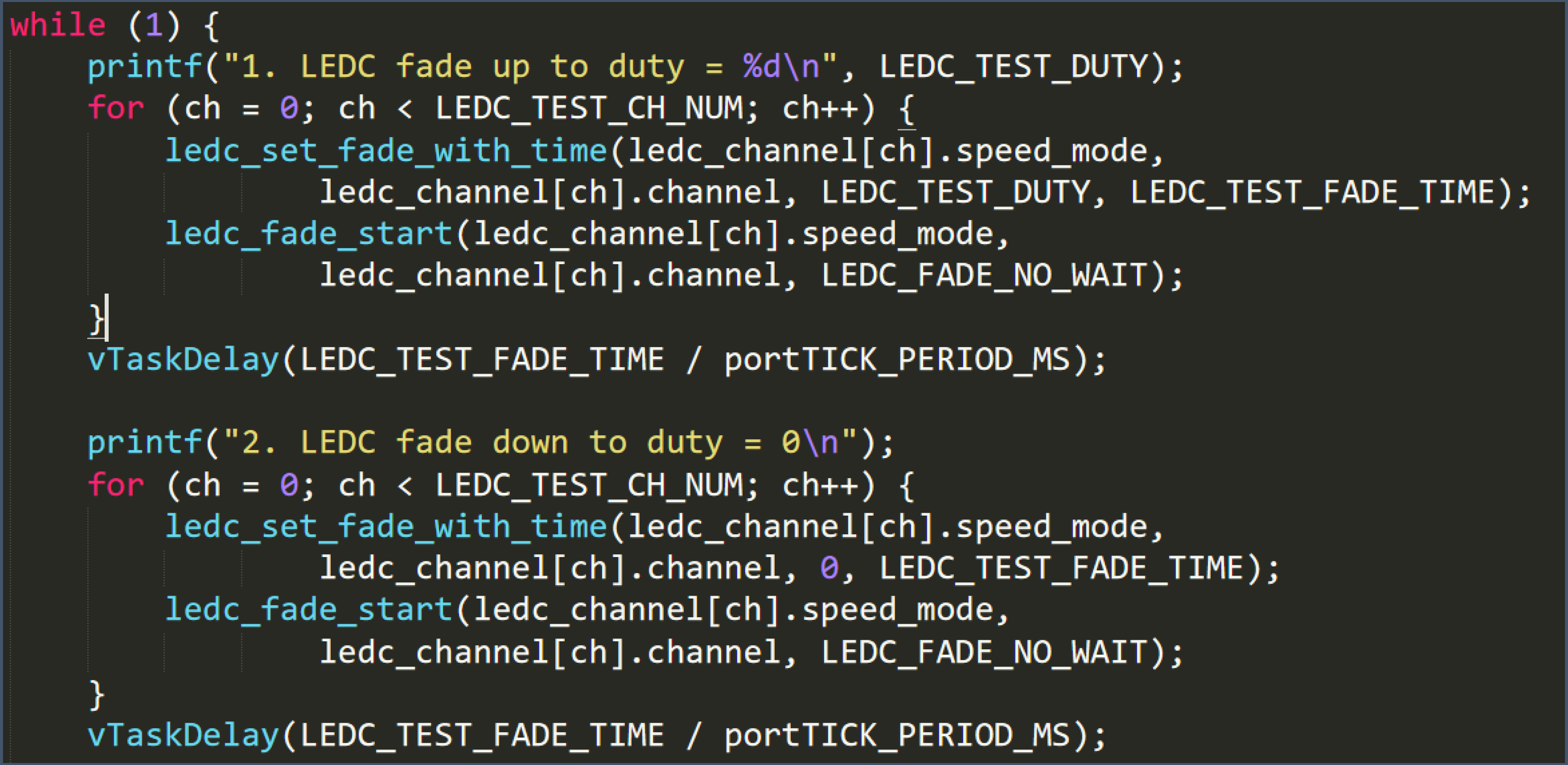
```c
ledc_timer_config_t ledc_timer = {
    .duty_resolution = LEDC_TIMER_13_BIT, // resolution of PWM duty
    .freq_hz = 5000,                      // frequency of PWM signal
    .speed_mode = LEDC_HS_MODE,           // timer mode
    .timer_num = LEDC_HS_TIMER,            // timer index
    .clk_cfg = LEDC_AUTO_CLK,              // Auto select the source clock
};
// Set configuration of timer0 for high speed channels
ledc_timer_config(&ledc_timer);
```



```c
ledc_channel_config_t ledc_channel[LEDC_TEST_CH_NUM] = {
    {
        .channel    = LEDC_HS_CH0_CHANNEL,
        .duty       = 0,
        .gpio_num   = LEDC_HS_CH0_GPIO,
        .speed_mode = LEDC_HS_MODE,
          .speed_mode = LEDC_LS_MODE,
        .hpoint     = 0,
        .timer_sel  = LEDC_HS_TIMER
          .timer_sel  = LEDC_LS_TIMER
    },
```

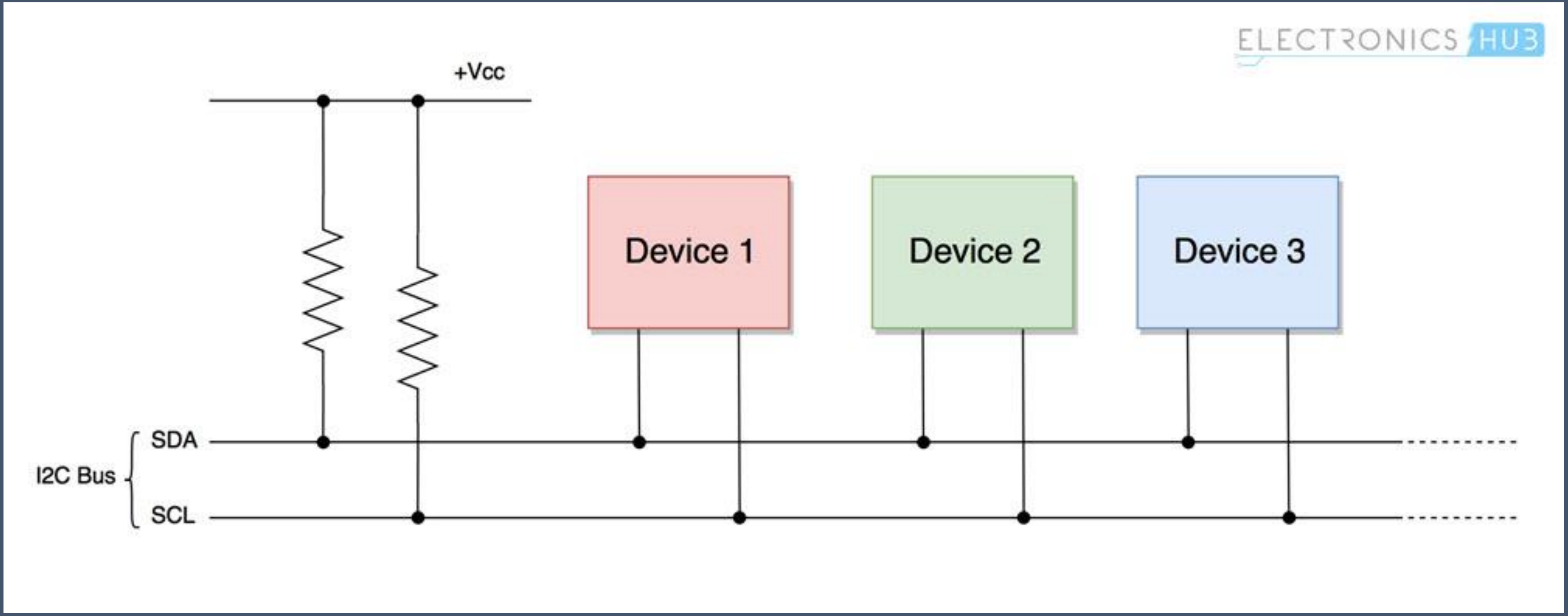


```c
// Set LED Controller with previously prepared configuration
for (ch = 0; ch < LEDC_TEST_CH_NUM; ch++) {
    ledc_channel_config(&ledc_channel[ch]);
}

// Initialize fade service.
ledc_fade_func_install(0);
```

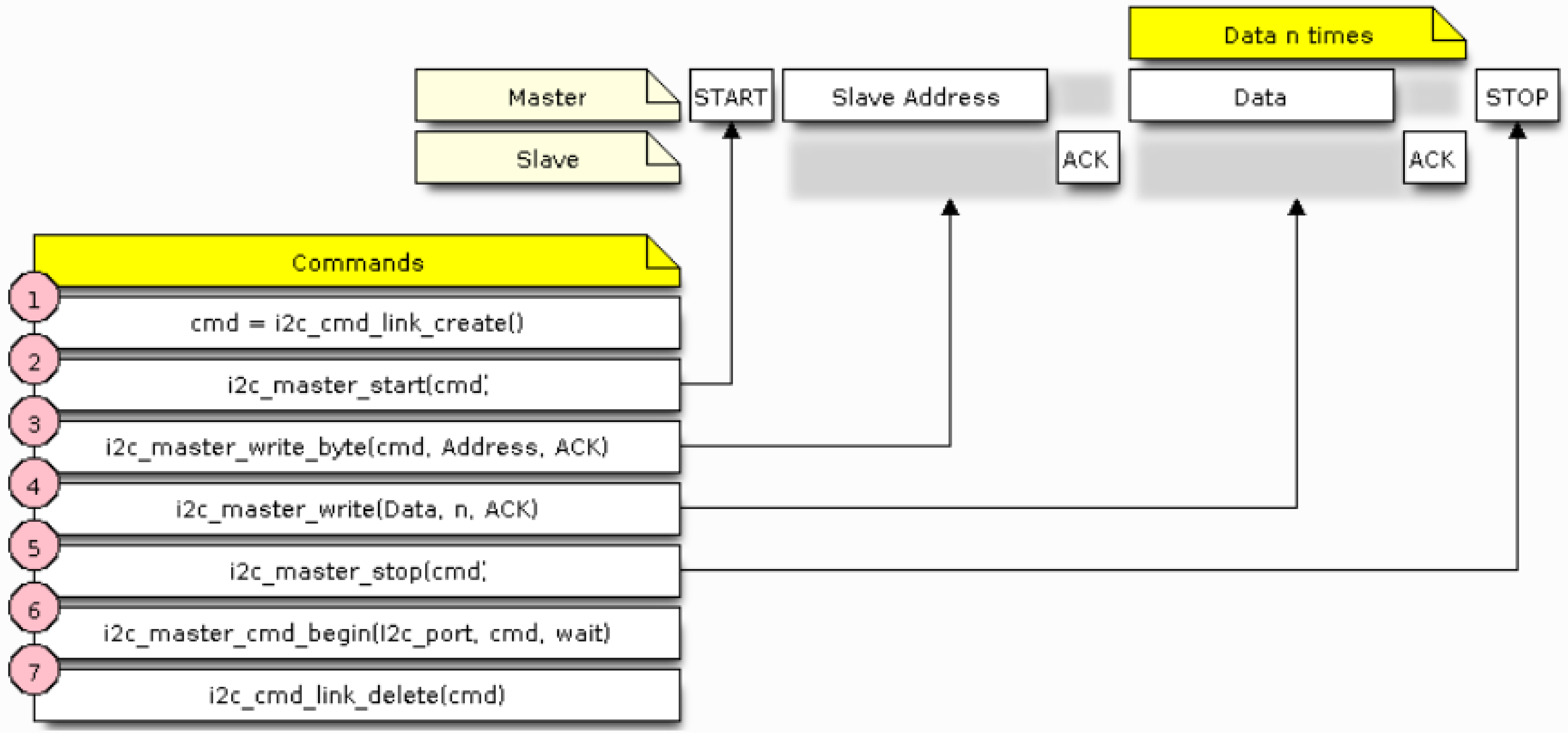


```c
while (1) {
    printf("1. LEDC fade up to duty = %d\n", LEDC_TEST_DUTY);
    for (ch = 0; ch < LEDC_TEST_CH_NUM; ch++) {
        ledc_set_fade_with_time(ledc_channel[ch].speed_mode,
                ledc_channel[ch].channel, LEDC_TEST_DUTY, LEDC_TEST_FADE_TIME);
        ledc_fade_start(ledc_channel[ch].speed_mode,
                ledc_channel[ch].channel, LEDC_FADE_NO_WAIT);
    }
    vTaskDelay(LEDC_TEST_FADE_TIME / portTICK_PERIOD_MS);

    printf("2. LEDC fade down to duty = 0\n");
    for (ch = 0; ch < LEDC_TEST_CH_NUM; ch++) {
        ledc_set_fade_with_time(ledc_channel[ch].speed_mode,
                ledc_channel[ch].channel, 0, LEDC_TEST_FADE_TIME);
        ledc_fade_start(ledc_channel[ch].speed_mode,
                ledc_channel[ch].channel, LEDC_FADE_NO_WAIT);
    }
    vTaskDelay(LEDC_TEST_FADE_TIME / portTICK_PERIOD_MS);
```

6. I2C

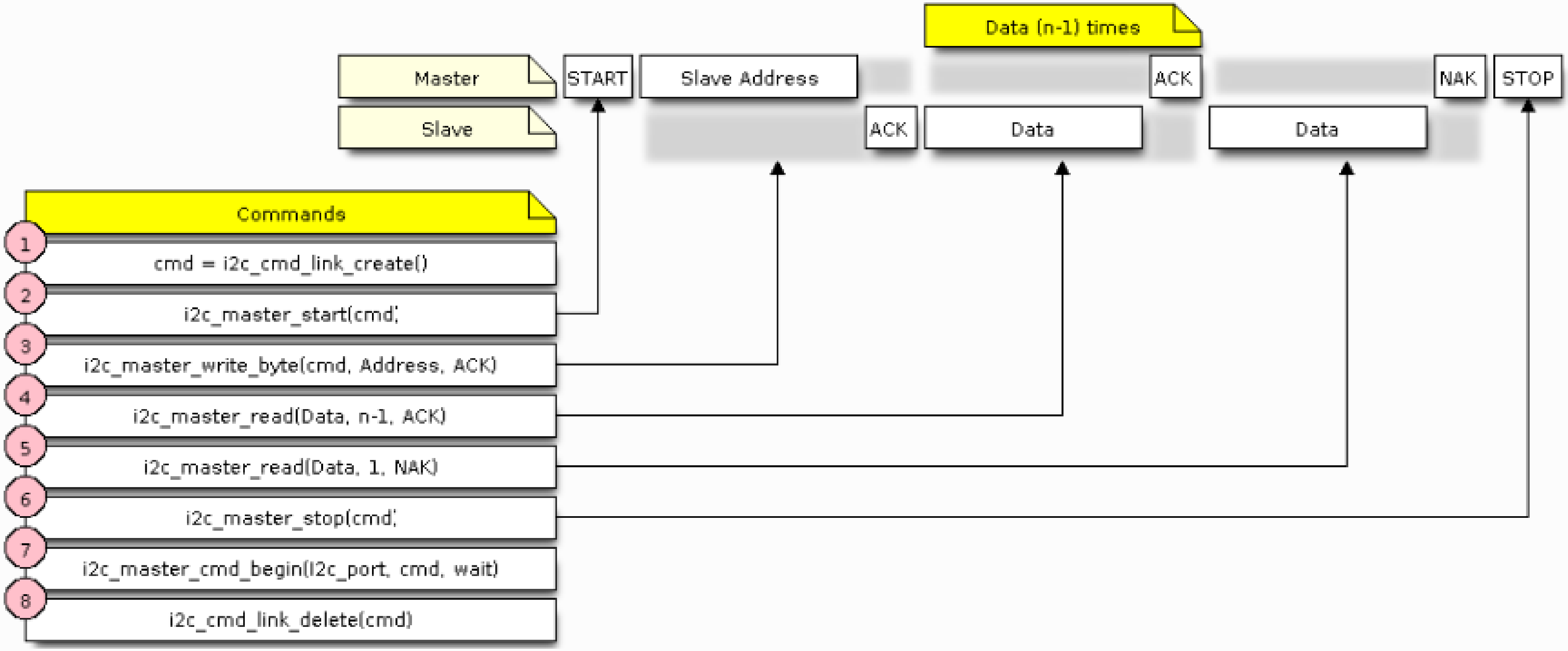


출처 : https://www.electronicshub.org/basics-i2c-communication/



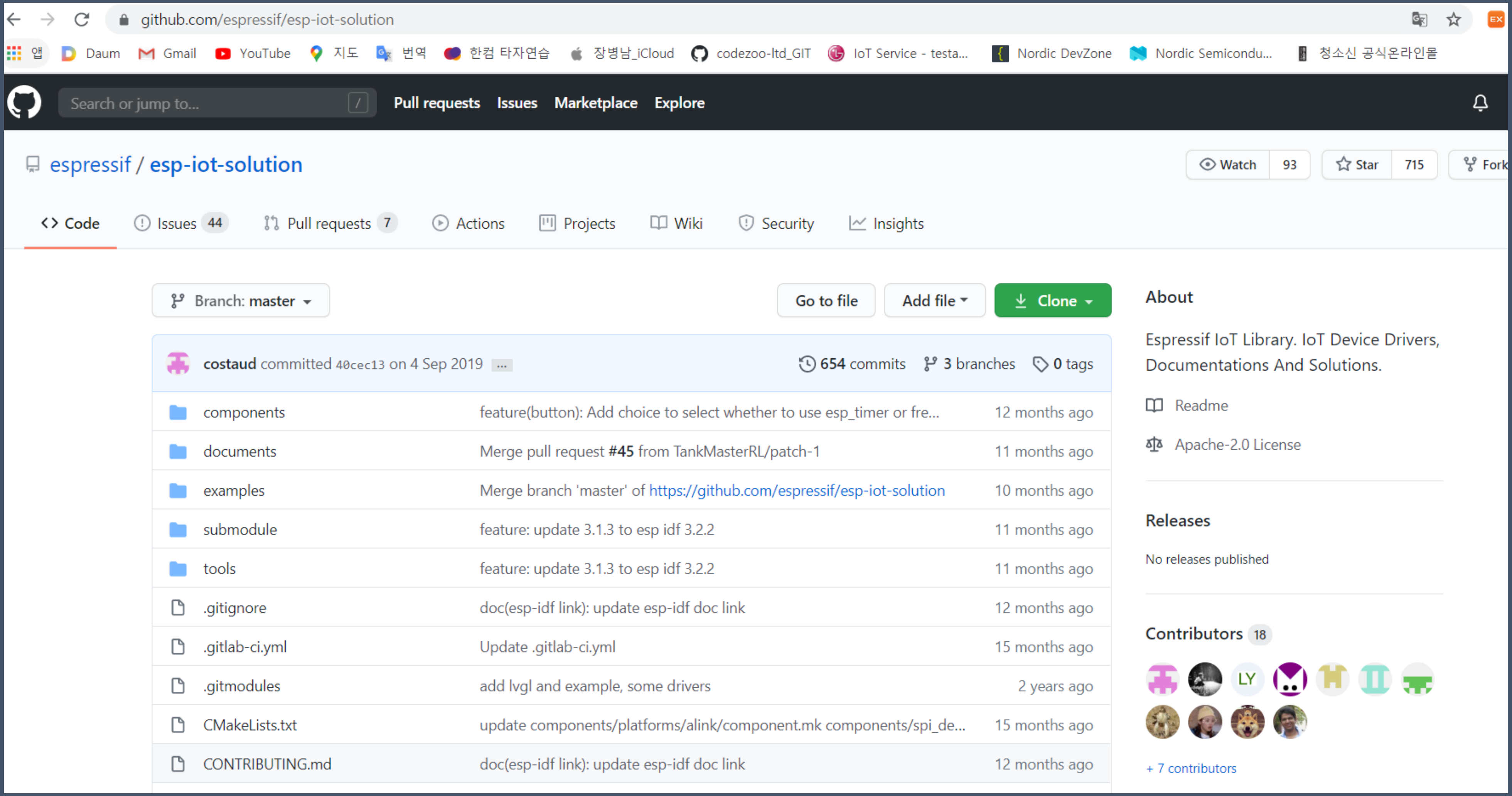
```
cmd = i2c_cmd_link_create()
i2c_master_start(cmd;
i2c_master_write_byte(cmd, Address, ACK)
i2c_master_write(Data, n, ACK)
i2c_master_stop(cmd;
i2c_master_cmd_begin(I2c_port, cmd, wait)
i2c_cmd_link_delete(cmd)
```

I2C 명령 링크- 마스터 쓰기 예


Data (n-1) times
Master
Slave
START
Slave Address
ACK
Data
ACK
Data
NAK
STOP
Commands
1
cmd = i2c_cmd_link_create()
2
i2c_master_start(cmd);
3
i2c_master_write_byte(cmd, Address, ACK)
4
i2c_master_read(Data, n-1, ACK)
5
i2c_master_read(Data, 1, NAK)
6
i2c_master_stop(cmd);
7
i2c_master_cmd_begin(I2c_port, cmd, wait)
8
i2c_cmd_link_delete(cmd)

I2C 명령 링크- 마스터 읽기 예

https://github.com/espressif/esp-iot-solution 에 올라간 소스 가져다가 포팅하기


github.com/espressif/esp-iot-solution

espressif / esp-iot-solution

Watch 93 Star 715 Fork

Code | Issues 44 | Pull requests 7 | Actions | Projects | Wiki | Security | Insights

Branch: master | Go to file | Add file | Clone

costaud committed 40cec13 on 4 Sep 2019 … 654 commits 3 branches 0 tags

components	feature(button): Add choice to select whether to use esp_timer or fre...	12 months ago
documents	Merge pull request #45 from TankMasterRL/patch-1	11 months ago
examples	Merge branch 'master' of https://github.com/espressif/esp-iot-solution	10 months ago
submodule	feature: update 3.1.3 to esp idf 3.2.2	11 months ago
tools	feature: update 3.1.3 to esp idf 3.2.2	11 months ago
.gitignore	doc(esp-idf link): update esp-idf doc link	12 months ago
.gitlab-ci.yml	Update .gitlab-ci.yml	15 months ago
.gitmodules	add lvgl and example, some drivers	2 years ago
CMakeLists.txt	update components/platforms/alink/component.mk components/spi_de...	15 months ago
CONTRIBUTING.md	doc(esp-idf link): update esp-idf doc link	12 months ago

About

Espressif IoT Library. IoT Device Drivers, Documentations And Solutions.

Readme

Apache-2.0 License

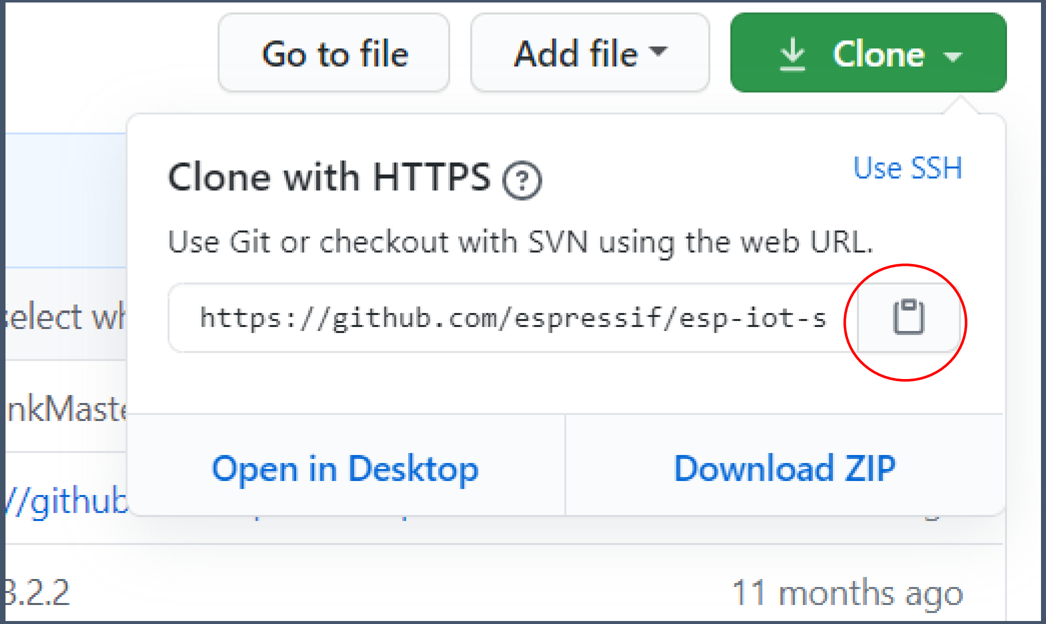
Releases

No releases published

Contributors 18

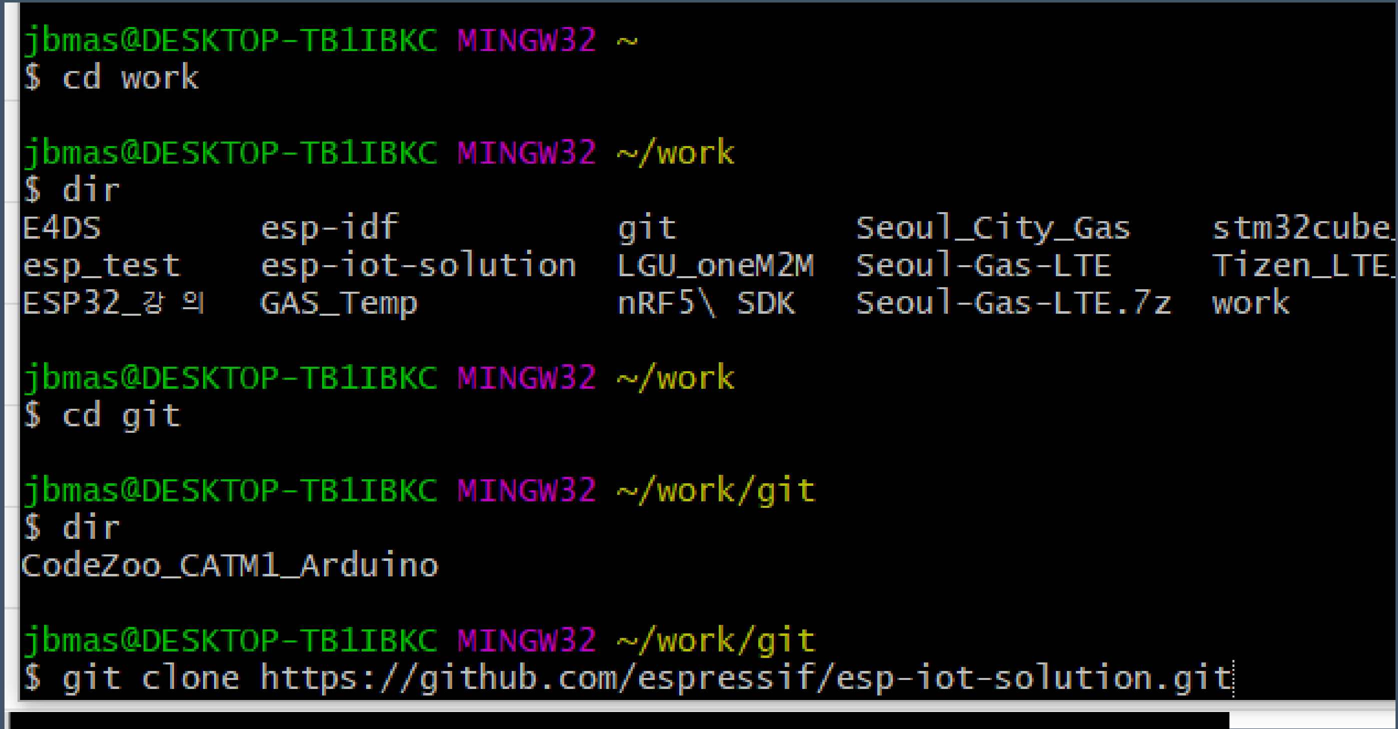
+ 7 contributors

https://github.com/espressif/esp-iot-solution 에 올라간 소스 가져다가 포팅하기

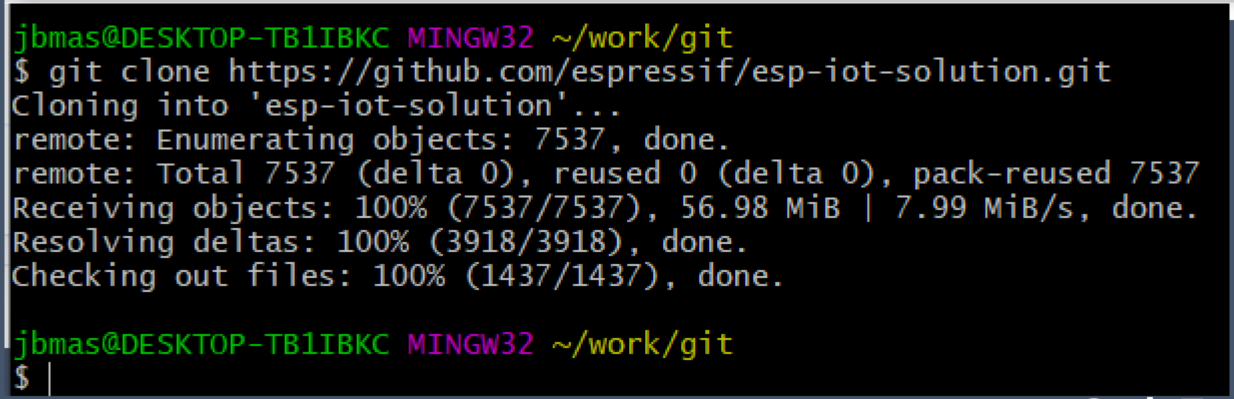


Use Git or checkout with SVN using the web URL.

```
https://github.com/espressif/esp-iot-s
```

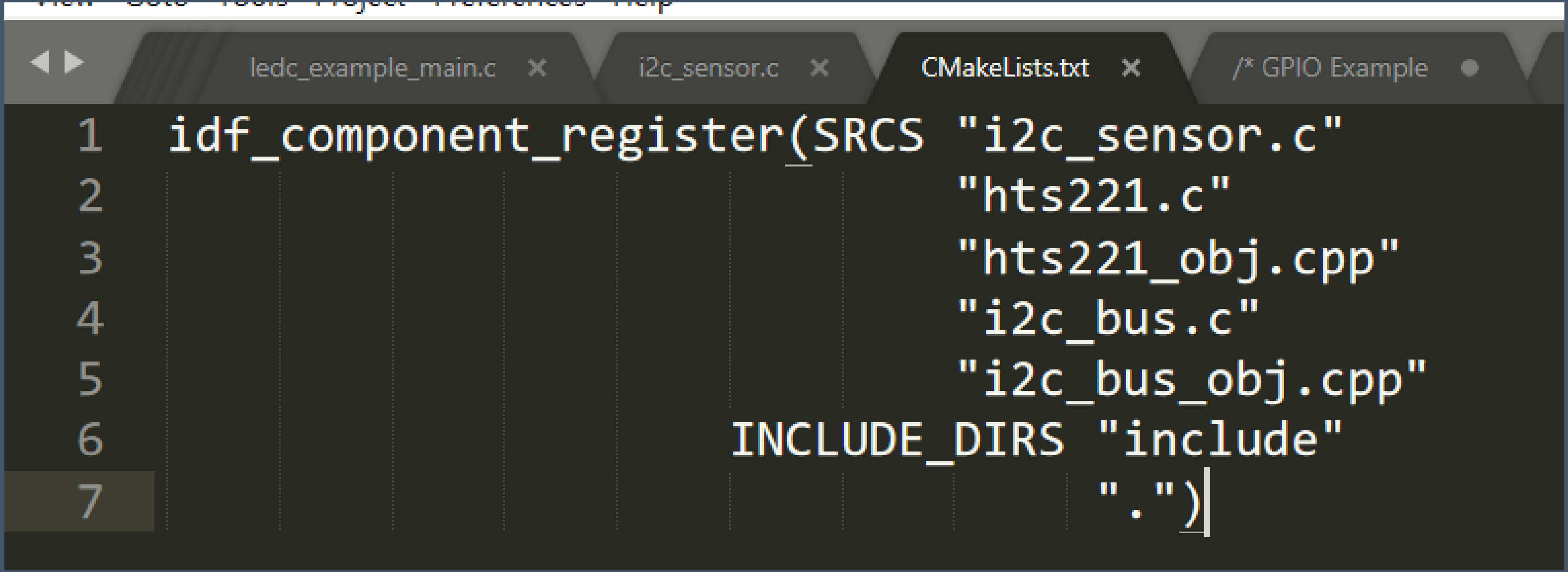


```
jbmas@DESKTOP-TB1IBKC MINGW32 ~
$ cd work

jbmas@DESKTOP-TB1IBKC MINGW32 ~/work
$ dir
E4DS         esp-idf            git         Seoul_City_Gas     stm32cube
esp_test     esp-iot-solution   LGU_oneM2M  Seoul-Gas-LTE      Tizen_LTE
ESP32_강 의   GAS_Temp           nRF5\ SDK   Seoul-Gas-LTE.7z   work

jbmas@DESKTOP-TB1IBKC MINGW32 ~/work
$ cd git

jbmas@DESKTOP-TB1IBKC MINGW32 ~/work/git
$ dir
CodeZoo_CATM1_Arduino

jbmas@DESKTOP-TB1IBKC MINGW32 ~/work/git
$ git clone https://github.com/espressif/esp-iot-solution.git
```

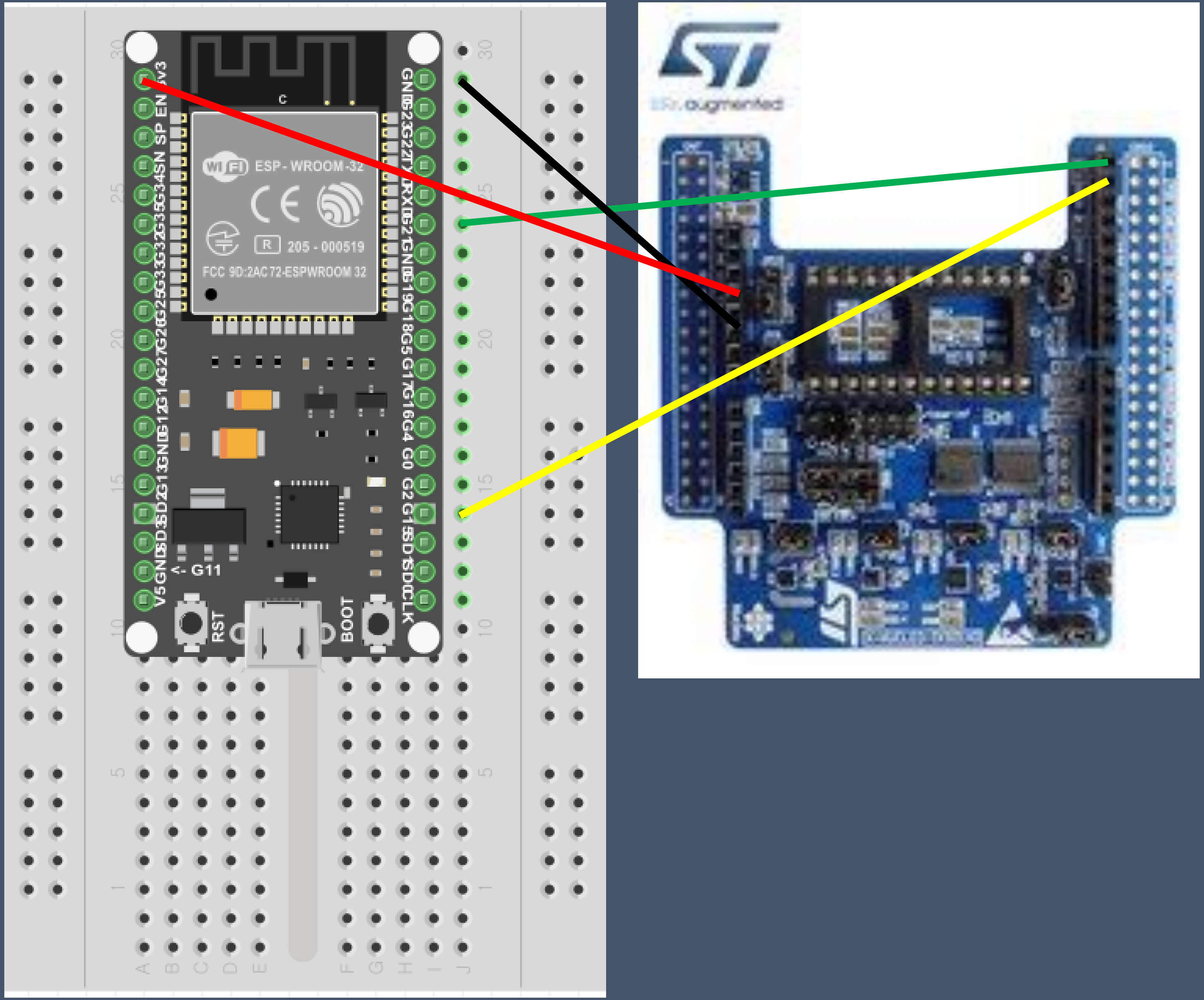


```
jbmas@DESKTOP-TB1IBKC MINGW32 ~/work/git
$ git clone https://github.com/espressif/esp-iot-solution.git
Cloning into 'esp-iot-solution'...
remote: Enumerating objects: 7537, done.
remote: Total 7537 (delta 0), reused 0 (delta 0), pack-reused 7537
Receiving objects: 100% (7537/7537), 56.98 MiB | 7.99 MiB/s, done.
Resolving deltas: 100% (3918/3918), done.
Checking out files: 100% (1437/1437), done.

jbmas@DESKTOP-TB1IBKC MINGW32 ~/work/git
$ 
```

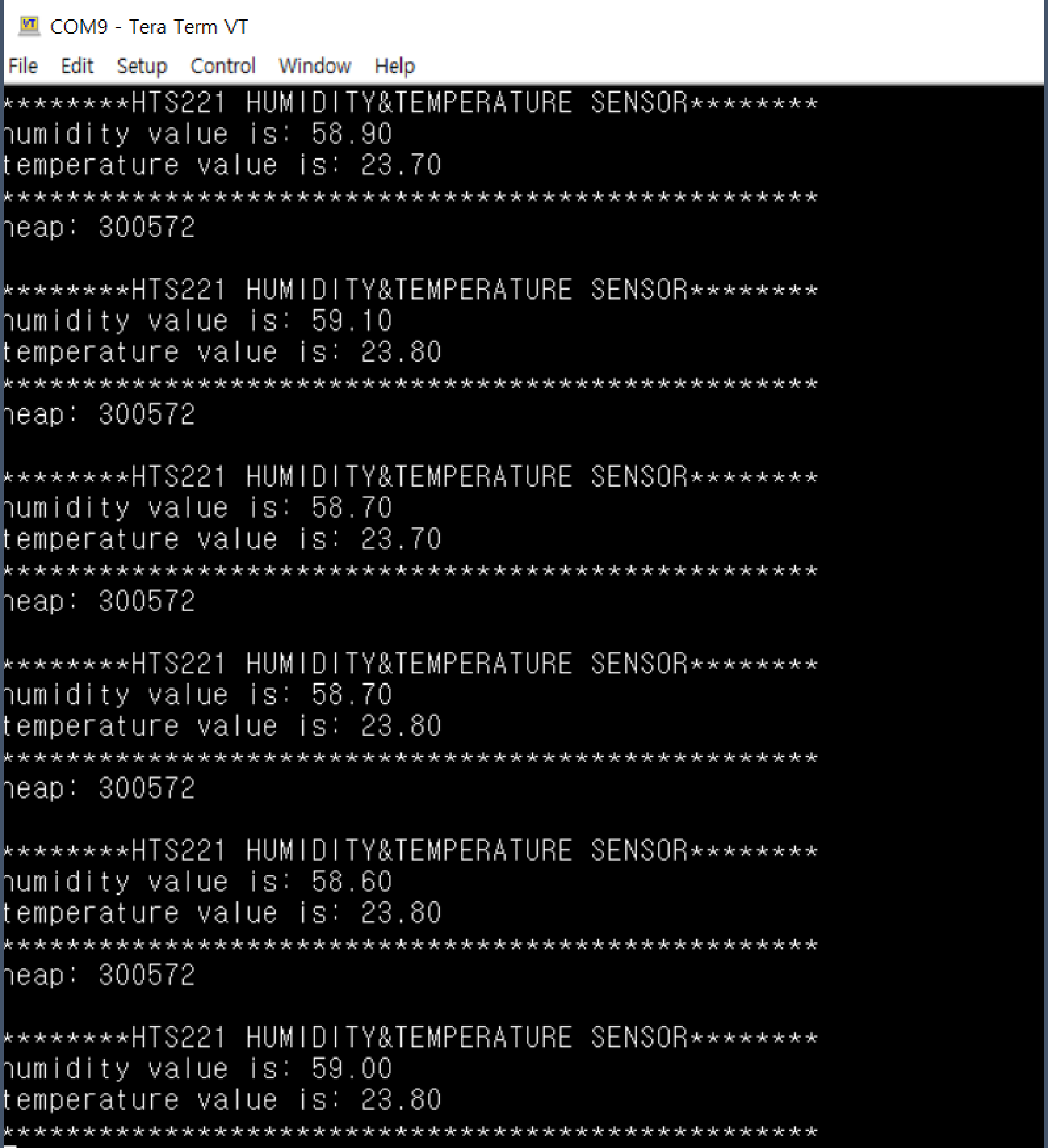
CMakeLists.txt 수정


ledc_example_main.c ×   i2c_sensor.c ×   CMakeLists.txt ×   /* GPIO Example ●

```
idf_component_register(SRCS "i2c_sensor.c"
                            "hts221.c"
                            "hts221_obj.cpp"
                            "i2c_bus.c"
                            "i2c_bus_obj.cpp"
                    INCLUDE_DIRS "include"
                                 ".")
```



GPIO21 -- SCL
GPIO15 -- SDA
GND -- GND
3.3V -- 3.3V



```
********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 58.90
temperature value is: 23.70
**************************************************
heap: 300572

********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 59.10
temperature value is: 23.80
**************************************************
heap: 300572

********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 58.70
temperature value is: 23.70
**************************************************
heap: 300572

********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 58.70
temperature value is: 23.80
**************************************************
heap: 300572

********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 58.60
temperature value is: 23.80
**************************************************
heap: 300572

********HTS221 HUMIDITY&TEMPERATURE SENSOR********
humidity value is: 59.00
temperature value is: 23.80
**************************************************
```

감사합니다.