DAC Func

DAC Func

Write a DAC example which have following features:

- generate the wave
- based on TIMx and Interrupt

initialize the timer.
write a isr interrupt, and put the wave generate func inside.

产生波形

```
    uint16_t DAC_data[];
    u8 key;
    void DAC_Channel1Init(void){
        DAC_Type()
    }
```

波形种类

```
void Gen_Wave(u8 wave_mode, short volt_max, u16 wave[]){
    short i = 0;
    float temp;
    short temp2;
    switch(wave_mode){
        case Wave_sin:
            temp = 2*3.1415926/Dot_x;
            for(i=0;i<Dot_x;i++){
                wave[i] = 0.5 * volt_max *(sin(temp*i)+1);
            }
            break;
        case Wave_Triangular:

    }
}
```

TIMER

Timer Initialization

> An ESP32 timer group should be identified using `timer_group_t`.

```
typedef enum {
    TIMER_GROUP_0 = 0, /*!<Hw timer group 0*/
    TIMER_GROUP_1 = 1, /*!<Hw timer group 1*/
    TIMER_GROUP_MAX,
} timer_group_t;
```

> An individual timer in a group should be identified with `timer_idx_t`.

```
typedef enum {
    TIMER_0 = 0, /*!<Select timer0 of GROUPx*/
    TIMER_1 = 1, /*!<Select timer1 of GROUPx*/
    TIMER_MAX,
} timer_idx_t;
```

The timer should be initialized by calling the function `timer_init()`.

```
esp_err_t esp_timer_init(void)
{
    esp_err_t err;
    if (is_initialized()) {
        return ESP_ERR_INVALID_STATE;
    }
#if CONFIG_SPIRAM_USE_MALLOC
    memset(&s_timer_semaphore_memory, 0, sizeof(StaticQueue_t));
    s_timer_semaphore = xSemaphoreCreateCountingStatic(TIMER_EVENT_QUEUE_SIZE, 0,
&s_timer_semaphore_memory);
#else
    s_timer_semaphore = xSemaphoreCreateCounting(TIMER_EVENT_QUEUE_SIZE, 0);
#endif
    if (!s_timer_semaphore) {
        err = ESP_ERR_NO_MEM;
        goto out;
    }
    int ret = xTaskCreatePinnedToCore(&timer_task, "esp_timer",
            ESP_TASK_TIMER_STACK, NULL, ESP_TASK_TIMER_PRIO, &s_timer_task, PRO_CPU_NUM);
    if (ret != pdPASS) {
        err = ESP_ERR_NO_MEM;
        goto out;
    }
    err = esp_timer_impl_init(&timer_alarm_handler);
    if (err != ESP_OK) {
        goto out;
    }
    return ESP_OK;
out:
    if (s_timer_task) {
        vTaskDelete(s_timer_task);
        s_timer_task = NULL;
    }
    if (s_timer_semaphore) {
        vSemaphoreDelete(s_timer_semaphore);
        s_timer_semaphore = NULL;
    }
    return ESP_ERR_NO_MEM;
}
```

//初始化ｔｉｍ，注册中断回调函数
//０．１ｕｍ

Passing a structure `timer_config_t` to it to define how the timer should operate.

```
typedef struct {
    bool alarm_en;      /*!< Timer alarm enable */
    bool counter_en;    /*!< Counter enable */
    timer_intr_mode_t intr_type; /*!< Interrupt mode */
    timer_count_dir_t counter_dir; /*!< Counter direction  */
    bool auto_reload;   /*!< Timer auto-reload */
    uint32_t divider;   /*!< Counter clock divider. The divider's range is from from 2 to 65536. */
} timer_config_t;
```

The following timer parameters can be set:

```
uint32_t divider;   /*!< Counter clock divider.The divider's range is from from 2 to 65536.Divider is
set how quickly the timer’s counter is “ticking”. The setting divider is used as a divisor of the
incoming 80 MHz APB_CLK clock. */

timer_count_dir_t counter_dir; /*!< Counter direction  */
/*typedef enum {
    TIMER_COUNT_DOWN = 0, // Descending Count from cnt.high|cnt.low
    TIMER_COUNT_UP = 1,   // Ascending Count from Zero
    TIMER_COUNT_MAX
} timer_count_dir_t;*/
```

Timer Control

Once the timer is enabled, its counter starts running.
To enable the timer, call the function `timer_init()` with `counter_en` set to true, or call `timer_start()`.

```
typedef enum {
    TIMER_0 = 0, /*!<Select timer0 of GROUPx*/
    TIMER_1 = 1, /*!<Select timer1 of GROUPx*/
    TIMER_MAX,
} timer_idx_t;
```

> You can specify the timer’s initial counter value by calling timer_set_counter_value()


```
typedef enum {
    TIMER_0 = 0, /*!<Select timer0 of GROUPx*/
    TIMER_1 = 1, /*!<Select timer1 of GROUPx*/
    TIMER_MAX,
} timer_idx_t;
```


> To check the timer’s current value, call timer_get_counter_value() or timer_get_counter_time_sec().


```
typedef enum {
    TIMER_0 = 0, /*!<Select timer0 of GROUPx*/
    TIMER_1 = 1, /*!<Select timer1 of GROUPx*/
    TIMER_MAX,
} timer_idx_t;
```


> Pause the timer at any time, call `timer_pause()`.
> To resume it, call `timer_start()`.

---

timer_group_example_main.c

```
/* Timer group-hardware timer example

   This example code is in the Public Domain (or CC0 licensed, at your option.)

   Unless required by applicable law or agreed to in writing, this
```

```
   software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
   CONDITIONS OF ANY KIND, either express or implied.
*/

#include <stdio.h>
#include "esp_types.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/periph_ctrl.h"
#include "driver/timer.h"

#define TIMER_DIVIDER         16  //  Hardware timer clock divider
#define TIMER_SCALE           (TIMER_BASE_CLK / TIMER_DIVIDER)  // convert counter value to seconds
#define TIMER_INTERVAL0_SEC   (3.4179) // sample test interval for the first timer
#define TIMER_INTERVAL1_SEC   (5.78)   // sample test interval for the second timer
#define TEST_WITHOUT_RELOAD   0        // testing will be done without auto reload
#define TEST_WITH_RELOAD      1        // testing will be done with auto reload

/*
 * 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;

xQueueHandle timer_queue;

/*
 * A simple helper function to print the raw timer counter value
 * and the counter value converted to seconds
 */
static void inline print_timer_counter(uint64_t counter_value)
{
    printf("Counter: 0x%08x%08x\n", (uint32_t) (counter_value >> 32),
                                    (uint32_t) (counter_value));
    printf("Time   : %.8f s\n", (double) counter_value / TIMER_SCALE);
}

/*
 * 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)
{
    int timer_idx = (int) para;

    /* Retrieve the interrupt status and the counter value
       from the timer that reported the interrupt */
    uint32_t intr_status = TIMERG0.int_st_timers.val;
    TIMERG0.hw_timer[timer_idx].update = 1;
    uint64_t timer_counter_value =
        ((uint64_t) TIMERG0.hw_timer[timer_idx].cnt_high) << 32
        | TIMERG0.hw_timer[timer_idx].cnt_low;

    /* 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 ((intr_status & BIT(timer_idx)) && timer_idx == TIMER_0) {
        evt.type = TEST_WITHOUT_RELOAD;
        TIMERG0.int_clr_timers.t0 = 1;
        timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
        TIMERG0.hw_timer[timer_idx].alarm_high = (uint32_t) (timer_counter_value >> 32);
        TIMERG0.hw_timer[timer_idx].alarm_low = (uint32_t) timer_counter_value;
    } else if ((intr_status & BIT(timer_idx)) && timer_idx == TIMER_1) {
        evt.type = TEST_WITH_RELOAD;
        TIMERG0.int_clr_timers.t1 = 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 */
    TIMERG0.hw_timer[timer_idx].config.alarm_en = TIMER_ALARM_EN;

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

/*
 * 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;
    config.divider = TIMER_DIVIDER;
    config.counter_dir = TIMER_COUNT_UP;
    config.counter_en = TIMER_PAUSE;
    config.alarm_en = TIMER_ALARM_EN;
    config.intr_type = TIMER_INTR_LEVEL;
    config.auto_reload = auto_reload;
    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);
}

/*
 * 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);
    }
}

/*
 * In this example, we will test hardware timer0 and timer1 of timer group0.
 */
void app_main()
{
```

```
    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);
}
```