/\* LEDC (LED Controller) fade 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 "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "driver/ledc.h"

#include "esp\_err.h"

/\*

\* About this example

\*

\* 1. Start with initializing LEDC module:

\* a. Set the timer of LEDC first, this determines the frequency

\* and resolution of PWM.

\* b. Then set the LEDC channel you want to use,

\* and bind with one of the timers.

\*

\* 2. You need first to install a default fade function,

\* then you can use fade APIs.

\*

\* 3. You can also set a target duty directly without fading.

\*

\* 4. This example uses GPIO18/19/4/5 as LEDC output,

\* and it will change the duty repeatedly.

\*

\* 5. GPIO18/19 are from high speed channel group.

\* GPIO4/5 are from low speed channel group.

\*

\*/

#define LEDC\_HS\_TIMER LEDC\_TIMER\_0

#define LEDC\_HS\_MODE LEDC\_HIGH\_SPEED\_MODE

#define LEDC\_HS\_CH0\_GPIO (18)

#define LEDC\_HS\_CH0\_CHANNEL LEDC\_CHANNEL\_0

#define LEDC\_HS\_CH1\_GPIO (19)

#define LEDC\_HS\_CH1\_CHANNEL LEDC\_CHANNEL\_1

#define LEDC\_LS\_TIMER LEDC\_TIMER\_1

#define LEDC\_LS\_MODE LEDC\_LOW\_SPEED\_MODE

#define LEDC\_LS\_CH2\_GPIO (4)

#define LEDC\_LS\_CH2\_CHANNEL LEDC\_CHANNEL\_2

#define LEDC\_LS\_CH3\_GPIO (5)

#define LEDC\_LS\_CH3\_CHANNEL LEDC\_CHANNEL\_3

#define LEDC\_TEST\_CH\_NUM (4)

#define LEDC\_TEST\_DUTY (4000)

#define LEDC\_TEST\_FADE\_TIME (3000)

//exit chars

#define CHAR\_ESC 0x1B

#define CHAR\_DEL 0x7F

#define CHAR\_BS 0x08

//enter char

#define CHAR\_ENTER '\n'

//other acceptable chars

#define CHAR\_SPACE 0x20

#define CHAR\_PERIOD 0x2E

#define CHAR\_COMMA 0x2C

#define CHAR\_SEMICOLON ':'

//getcahr null return

#define CHAR\_NULL 0xFF

//delay defines

#define INPUT\_DELAY 33 / portTICK\_PERIOD\_MS

#define DMX\_DELAY 3

//size defines

#define INPUT\_BUFFER\_SIZE 64

void app\_main()

{

char in;

uint32\_t freq = 500;

uint32\_t duty = 0;

ledc\_timer\_config\_t ledc\_timer = {

.duty\_resolution = LEDC\_TIMER\_13\_BIT, // resolution of PWM duty

.freq\_hz = freq, // frequency of PWM signal

.speed\_mode = LEDC\_HS\_MODE, // timer mode

.timer\_num = LEDC\_HS\_TIMER // timer index

};

ledc\_timer\_config(&ledc\_timer);

ledc\_channel\_config\_t ledc\_channel =

{

.channel = LEDC\_HS\_CH0\_CHANNEL,

.duty = 0,

.gpio\_num = LEDC\_HS\_CH0\_GPIO,

.speed\_mode = LEDC\_HS\_MODE,

.hpoint = 0,

.timer\_sel = LEDC\_HS\_TIMER

};

ledc\_channel\_config(&ledc\_channel);

while(1){

in = getchar();

if(in == CHAR\_DEL || in == CHAR\_BS){

putchar('\n');

continue;

}

switch(in){

case 'U':

if(freq + 100 <= 5000){

freq += 100;

ledc\_set\_freq(ledc\_channel.speed\_mode, ledc\_channel.timer\_sel, freq);

}

printf("freq up %d\n", freq);

break;

case 'D':

if(freq - 100 > 100){

freq -= 100;

ledc\_set\_freq(ledc\_channel.speed\_mode, ledc\_channel.timer\_sel, freq);

}

printf("freq down %d\n", freq);

break;

case 'u':

if(duty + 1 <= 8191){

duty += 1;

ledc\_set\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel, duty);

ledc\_update\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel);

}

printf("duty up %d\n", duty);

break;

case 'd':

if((duty - 1) < 8191){

duty -= 1;

ledc\_set\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel, duty);

ledc\_update\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel);

} else {

duty = 0;

ledc\_set\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel, duty);

ledc\_update\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel);

}

printf("duty down %d\n", duty);

break;

case 'Z':

case 'z':

duty = 0;

ledc\_set\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel, duty);

ledc\_update\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel);

printf("duty zero %d\n", duty);

break;

case 'F':

duty = 8191;

ledc\_set\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel, duty);

ledc\_update\_duty(ledc\_channel.speed\_mode, ledc\_channel.channel);

printf("duty full %d\n", duty);

break;

break;

}

vTaskDelay(1);

}

}