LED Control (LEDC)

About

The LED control (LEDC) peripheral is primarly designed to control the intensity of LEDs, although it can also be used to generate PWM signals for other purposes.

ESP32 SoCs has from 6 to 16 channels (variates on socs, see table below) which can generate independent waveforms, that can be used for example to drive RGB LED devices.

ESP32 SoC	Number of LEDC channels
ESP32	16
ESP32-S2	8
ESP32-S3	8
ESP32-C3	6
ESP32-C6	6
ESP32-H2	6

Arduino-ESP32 LEDC API

ledcAttach

This function is used to setup LEDC pin with given frequency and resolution. LEDC channel will be selected automatically.

```
bool ledcAttach(uint8_t pin, uint32_t freq, uint8_t resolution);
```

- `pin` select LEDC pin.
- `freq` select frequency of pwm.
- `resolution` select resolution for LEDC channel.
  - range is 1-14 bits (1-20 bits for ESP32).

This function will return `true` if configuration is successful. If `false` is returned, error occurs and LEDC channel was not configured.

ledcAttachChannel

This function is used to setup LEDC pin with given frequency, resolution and channel.

```
bool ledcAttachChannel(uint8_t pin, uint32_t freq, uint8_t resolution, uint8_t channel);
```

- `pin` select LEDC pin.
- `freq` select frequency of pwm.
- `resolution` select resolution for LEDC channel.
- `channel` select LEDC channel.
  - range is 1-14 bits (1-20 bits for ESP32).

This function will return `true` if configuration is successful. If `false` is returned, error occurs and LEDC channel was not configured.

ledcWrite

This function is used to set duty for the LEDC pin.

```
void ledcWrite(uint8_t pin, uint32_t duty);
```

- `pin` select LEDC pin.
- `duty` select duty to be set for selected LEDC pin.

This function will return `true` if setting duty is successful. If `false` is returned, error occurs and duty was not set.

ledcRead

This function is used to get configured duty for the LEDC pin.

```
uint32_t ledcRead(uint8_t pin);
```

- `pin` select LEDC pin to read the configured LEDC duty.

This function will return `duty` set for selected LEDC pin.

ledcReadFreq

This function is used to get configured frequency for the LEDC channel.

```
uint32_t ledcReadFreq(uint8_t pin);
```

- `pin` select LEDC pin to read the configured frequency.

This function will return `frequency` configured for selected LEDC pin.

ledcWriteTone

This function is used to setup the LEDC pin to 50 % PWM tone on selected frequency.

```
uint32_t ledcWriteTone(uint8_t pin, uint32_t freq);
```

- `pin` select LEDC pin.
- `freq` select frequency of pwm signal. If frequency is `0`, duty will be set to 0.

This function will return `frequency` set for LEDC pin. If `0` is returned, error occurs and LEDC pin was not configured.

ledcWriteNote

This function is used to setup the LEDC pin to specific note.

```
uint32_t ledcWriteNote(uint8_t pin, note_t note, uint8_t octave);
```

- `pin` select LEDC pin.
- `note` select note to be set.

NOTE_C	NOTE_Cs	NOTE_D	NOTE_Eb	NOTE_E	NOTE_F
NOTE_Fs	NOTE_G	NOTE_Gs	NOTE_A	NOTE_Bb	NOTE_B

- `octave` select octave for note.

This function will return `frequency` configured for the LEDC pin according to note and octave inputs. If `0` is returned, error occurs and the LEDC channel was not configured.

ledcDetach

This function is used to detach the pin from LEDC.

```
bool ledcDetach(uint8_t pin);
```

- `pin` select LEDC pin.

This function returns `true` if detaching was successful. If `false` is returned, an error occurred and the pin was not detached.

ledcChangeFrequency

This function is used to set frequency for the LEDC pin.

```
uint32_t ledcChangeFrequency(uint8_t pin, uint32_t freq, uint8_t resolution);
```

- `pin` select LEDC pin.
- `freq` select frequency of pwm.
- `resolution` select resolution for LEDC channel.
  - range is 1-14 bits (1-20 bits for ESP32).

This function will return `frequency` configured for the LEDC channel. If `0` is returned, error occurs and the LEDC channel frequency was not set.

ledcFade

This function is used to setup and start fade for the LEDC pin.

```
bool ledcFade(uint8_t pin, uint32_t start_duty, uint32_t target_duty, int max_fade_time_ms);
```

- `pin` select LEDC pin.
- `start_duty` select starting duty of fade.
- `target_duty` select target duty of fade.
- `max_fade_time_ms` select maximum time for fade.

This function will return `true` if configuration is successful. If `false` is returned, error occurs and LEDC fade was not configured / started.

ledcFadeWithInterrupt

This function is used to setup and start fade for the LEDC pin with interrupt.

```
bool ledcFadeWithInterrupt(uint8_t pin, uint32_t start_duty, uint32_t target_duty, int
max_fade_time_ms, void (*userFunc)(void));
```

- `pin` select LEDC pin.
- `start_duty` select starting duty of fade.
- `target_duty` select target duty of fade.
- `max_fade_time_ms` select maximum time for fade.
- `userFunc` funtion to be called when interrupt is triggered.

This function will return `true` if configuration is successful and fade start. If `false` is returned, error occurs and LEDC fade was not configured / started.

ledcFadeWithInterruptArg

This function is used to setup and start fade for the LEDC pin with interrupt using arguments.

```
bool ledcFadeWithInterruptArg(uint8_t pin, uint32_t start_duty, uint32_t target_duty, int
max_fade_time_ms, void (*userFunc)(void*), void * arg);
```

- `pin` select LEDC pin.
- `start_duty` select starting duty of fade.
- `target_duty` select target duty of fade.
- `max_fade_time_ms` select maximum time for fade.
- `userFunc` funtion to be called when interrupt is triggered.
- `arg` pointer to the interrupt arguments.

This function will return `true` if configuration is successful and fade start. If `false` is returned, error occurs and LEDC fade was not configured / started.

analogWrite

This function is used to write an analog value (PWM wave) on the pin. It is compatible with Arduinos analogWrite function.

```
void analogWrite(uint8_t pin, int value);
```

- `pin` select the GPIO pin.
- `value` select the duty cycle of pwm. * range is from 0 (always off) to 255 (always on).

analogWriteResolution

This function is used to set resolution for selected analogWrite pin.

```
void analogWriteResolution(uint8_t pin, uint8_t resolution);
```

- `pin` select the GPIO pin.
- `resolution` select resolution for analog channel.

analogWriteFrequency

This function is used to set frequency for selected analogWrite pin.

```
void analogWriteFrequency(uint8_t pin, uint32_t freq);
```

- `pin` select the GPIO pin.
- `freq` select frequency of pwm.

Example Applications

LEDC fade example:

```
/* LEDC Fade Arduino 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.
*/

// use 12 bit precission for LEDC timer
#define LEDC_TIMER_12_BIT  12

// use 5000 Hz as a LEDC base frequency
#define LEDC_BASE_FREQ     5000

// fade LED PIN (replace with LED_BUILTIN constant for built-in LED)
#define LED_PIN            4

// define starting duty, target duty and maximum fade time
#define LEDC_START_DUTY   (0)
#define LEDC_TARGET_DUTY  (4095)
#define LEDC_FADE_TIME    (3000)

bool fade_ended = false;      // status of LED fade
bool fade_on = true;

void ARDUINO_ISR_ATTR LED_FADE_ISR()
{
  fade_ended = true;
}

void setup() {
  // Initialize serial communication at 115200 bits per second:
  Serial.begin(115200);
  while(!Serial) delay(10);

  // Setup timer and attach timer to a led pins
  ledcAttach(LED_PIN, LEDC_BASE_FREQ, LEDC_TIMER_12_BIT);

  // Setup and start fade on led (duty from 0 to 4095)
  ledcFade(LED_PIN, LEDC_START_DUTY, LEDC_TARGET_DUTY, LEDC_FADE_TIME);
  Serial.println("LED Fade on started.");

  // Wait for fade to end
  delay(LEDC_FADE_TIME);

  // Setup and start fade off led and use ISR (duty from 4095 to 0)
  ledcFadeWithInterrupt(LED_PIN, LEDC_TARGET_DUTY, LEDC_START_DUTY, LEDC_FADE_TIME,
LED_FADE_ISR);
  Serial.println("LED Fade off started.");
}

void loop() {
  // Check if fade_ended flag was set to true in ISR
  if(fade_ended){
    Serial.println("LED fade ended");
    fade_ended = false;

    // Check if last fade was fade on
    if(fade_on){
      ledcFadeWithInterrupt(LED_PIN, LEDC_START_DUTY, LEDC_TARGET_DUTY, LEDC_FADE_TIME,
LED_FADE_ISR);
      Serial.println("LED Fade off started.");
      fade_on = false;
```

```
    }
    else {
      ledcFadeWithInterrupt(LED_PIN, LEDC_TARGET_DUTY, LEDC_START_DUTY, LEDC_FADE_TIME,
LED_FADE_ISR);
      Serial.println("LED Fade on started.");
      fade_on = true;
    }
  }
}
```

LEDC software fade example:

```
/*
 LEDC Software Fade

 This example shows how to software fade LED
 using the ledcWrite function.

 Code adapted from original Arduino Fade example:
 https://www.arduino.cc/en/Tutorial/Fade

 This example code is in the public domain.
 */

// use 12 bit precission for LEDC timer
#define LEDC_TIMER_12_BIT  12

// use 5000 Hz as a LEDC base frequency
#define LEDC_BASE_FREQ     5000

// fade LED PIN (replace with LED_BUILTIN constant for built-in LED)
#define LED_PIN            5

int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// Arduino like analogWrite
// value has to be between 0 and valueMax
void ledcAnalogWrite(uint8_t pin, uint32_t value, uint32_t valueMax = 255) {
  // calculate duty, 4095 from 2 ^ 12 - 1
  uint32_t duty = (4095 / valueMax) * min(value, valueMax);

  // write duty to LEDC
  ledcWrite(pin, duty);
}

void setup() {
  // Setup timer and attach timer to a led pin
  ledcAttach(LED_PIN, LEDC_BASE_FREQ, LEDC_TIMER_12_BIT);
}

void loop() {
  // set the brightness on LEDC channel 0
  ledcAnalogWrite(LED_PIN, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

LEDC Write RGB example:

```
/*
  ledcWrite_RGB.ino
  Runs through the full 255 color spectrum for an rgb led
  Demonstrate ledcWrite functionality for driving leds with PWM on ESP32

  This example code is in the public domain.

  Some basic modifications were made by vseven, mostly commenting.
 */

// Set up the rgb led names
uint8_t ledR = 0;
uint8_t ledG = 2;
uint8_t ledB = 4;

const boolean invert = true; // set true if common anode, false if common cathode

uint8_t color = 0;          // a value from 0 to 255 representing the hue
uint32_t R, G, B;           // the Red Green and Blue color components
uint8_t brightness = 255;  // 255 is maximum brightness, but can be changed.  Might need 256
for common anode to fully turn off.

// the setup routine runs once when you press reset:
void setup()
{
  Serial.begin(115200);
  delay(10);

  // Initialize pins as LEDC channels
  // resolution 1-16 bits, freq limits depend on resolution
  ledcAttach(ledR, 12000, 8); // 12 kHz PWM, 8-bit resolution
  ledcAttach(ledG, 12000, 8);
  ledcAttach(ledB, 12000, 8);
}

// void loop runs over and over again
void loop()
{
  Serial.println("Send all LEDs a 255 and wait 2 seconds.");
  // If your RGB LED turns off instead of on here you should check if the LED is common anode
or cathode.
  // If it doesn't fully turn off and is common anode try using 256.
  ledcWrite(ledR, 255);
  ledcWrite(ledG, 255);
  ledcWrite(ledB, 255);
  delay(2000);
  Serial.println("Send all LEDs a 0 and wait 2 seconds.");
  ledcWrite(ledR, 0);
  ledcWrite(ledG, 0);
  ledcWrite(ledB, 0);
  delay(2000);

  Serial.println("Starting color fade loop.");

 for (color = 0; color < 255; color++) { // Slew through the color spectrum

  hueToRGB(color, brightness);  // call function to convert hue to RGB

  // write the RGB values to the pins
  ledcWrite(ledR, R); // write red component to channel 1, etc.
  ledcWrite(ledG, G);
  ledcWrite(ledB, B);
```

```
  delay(100); // full cycle of rgb over 256 colors takes 26 seconds
 }

}

// Courtesy http://www.instructables.com/id/How-to-Use-an-RGB-LED/?ALLSTEPS
// function to convert a color to its Red, Green, and Blue components.

void hueToRGB(uint8_t hue, uint8_t brightness)
{
    uint16_t scaledHue = (hue * 6);
    uint8_t segment = scaledHue / 256; // segment 0 to 5 around the
                                       // color wheel
    uint16_t segmentOffset =
      scaledHue - (segment * 256); // position within the segment

    uint8_t complement = 0;
    uint16_t prev = (brightness * ( 255 -  segmentOffset)) / 256;
    uint16_t next = (brightness *  segmentOffset) / 256;

    if(invert)
    {
      brightness = 255 - brightness;
      complement = 255;
      prev = 255 - prev;
      next = 255 - next;
    }

    switch(segment ) {
    case 0:      // red
        R = brightness;
        G = next;
        B = complement;
    break;
    case 1:     // yellow
        R = prev;
        G = brightness;
        B = complement;
    break;
    case 2:     // green
        R = complement;
        G = brightness;
        B = next;
    break;
    case 3:    // cyan
        R = complement;
        G = prev;
        B = brightness;
    break;
    case 4:    // blue
        R = next;
        G = complement;
        B = brightness;
    break;
   case 5:      // magenta
    default:
        R = brightness;
        G = complement;
        B = prev;
    break;
    }
}
```